HAL
open science

Logical theories and abstract argumentation: A survey of existing works

Philippe Besnard, Claudette Cayrol, Marie-Christine Lagasquie-Schiex

To cite this version:

Philippe Besnard, Claudette Cayrol, Marie-Christine Lagasquie-Schiex. Logical theories and abstract argumentation: A survey of existing works. Argument and Computation, 2020, 11 (1-2), pp.41-102. 10.3233/AAC-190476 . hal-02875289

HAL Id: hal-02875289

https://hal.science/hal-02875289v1

Submitted on 1 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Open Archive Toulouse Archive Ouverte

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible

This is an author's version published in: https://oatao.univ-toulouse.fr/26425

Official URL :
https://doi.org/10.3233/AAC-190476

To cite this version:

Besnard, Philippe and Cayrol, Claudette and Lagasquié-Schiex, Marie-Christine *Logical theories and abstract argumentation: A survey of existing works.* (2020) Argument and Computation, 11 (1-2). 41-102. ISSN 1946-2166

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

Logical theories and abstract argumentation: A survey of existing works

Philippe Besnard [a], Claudette Cayrol [b] and Marie-Christine Lagasquie-Schiex [b,*]
[a] *IRIT, CNRS, Toulouse, France*
E-mail: philippe.besnard@irit.fr

[b] *IRIT, Toulouse 3 University, Toulouse, France*
E-mails: claudette.cayrol@irit.fr, marie-christine.lagasquie@irit.fr

Abstract. In 1995, in his seminal paper introducing the abstract argumentation framework, Dung has also established the first relationship between this framework and a logical framework (in this case: logic programming). Since that time, a lot of work have pursued this path, proposing different definitions, uses and exhibiting distinct relationships between argumentation and logic. In this paper, we present a survey of existing works about this topic and more especially those that address the following question: "How logic has been used for capturing various aspects or parts of Dung's argumentation". This survey covers many different approaches but is not intended to be totally exhaustive due to the huge quantity of papers in this scope. Moreover, due to the fact that each approach has its own specificities, sometimes antagonistic with the other approaches, and is also justified by its own context of definition or use, the aim of this survey is not to identify one approach as being better than another.

Keywords: Abstract argumentation, logics

1. Introduction

Argumentation has become an essential paradigm for Knowledge Representation and, especially, for reasoning from contradictory information [2,54] and for formalizing the exchange of arguments between agents in, e.g., negotiation [3]. Two main classes of approaches exist in computational argumentation: structured argumentation and abstract argumentation. The difference between these classes is on their starting point: works corresponding to the first class are about building arguments and identifying their relationships, whereas works in the second class consider that a collection of arguments interacting with each other is given [54], thereby disregarding the topic of building altogether. Because of these "abstract inputs", abstract frameworks have greatly eased the modelling and study of argumentation, allowing to focus, in particular, on the different ways for determining "acceptable" sets of arguments called *extensions*.

The relationship between abstract argumentation and logic has actually been exploited right from the start, in the seminal article [54] introducing abstract argumentation where Dung establishes a formal equivalence between argumentation frameworks and logic programs. Since this work, this relationship has been the theme of intensive research, and several correspondences between abstract argumentation and different logical theories have been established. The number of works in this topic is so huge that there is a critical need to clarify the links between all these works and that is the main aim of our paper.

*Corresponding author. E-mail: marie-christine.lagasquie@irit.fr.

So, this article is a survey about abstract argumentation papers that have a strong emphasis on logic. Two categories of approaches can be identified: (1) approaches taking advantage of logic to capture various aspects of abstract argumentation; (2) approaches embedding "logical structures" into abstract argumentation.[1]

For this paper, considering that this survey is published on the occasion of the 25 years of Dung's approach, we have chosen to discuss only works related to category (1) for the following reason: this a well-delimited category whereas the second one is less clearly defined; in particular, one can found in the second category some approaches also related to the structured argumentation and not only to the abstract argumentation.

Focussing on the many papers that provide an answer to the question "How can logic be used for achieving abstract argumentation?", we attempt to follow the same pattern for each work reviewed:[2]

- Since at the heart of these works is a correspondence between argumentation frameworks and logical formulae (or sets thereof), we first identify the entry points of the correspondence. Indeed, while the main input is usually an argumentation graph, some approaches handle an extended argumentation graph, be it bipolar, recursive, weighted, with collective interactions, and so on. There may also be extra inputs, for instance, special requirements and constraints, a distinguished subset of the arguments in the graph, a given argumentative semantics.
- Second, the aim of the approach reviewed is explicited, whether is it to provide a logical encoding of an argumentation graph, or to answer a question such as "is this subset of the arguments a preferred extension of the argumentation graph?".
- Third, we have a look at the type of logic employed in the work reviewed. Propositional logic is the most widely used, either as such or extended to Quantified Boolean Formulae, but some approaches resort to (possibly many-sorted) predicate logic, modal logics, as well as constructive logics (including intuitionistic logic) either directly or through a theoretical account of logic programming.
- Fourth, we list the auxiliary items involved in the approach at hand, e.g., labellings, signed atoms.
- Fifth, we deal with the question: Has the work reviewed been implemented?

In addition, the main topic in each case is of course about the role of logic in the work reviewed.

We now present the content of our survey in more detail. Section 2 gives the necessary background about abstract argumentation [6,28,31,54,81]. Then the presented works are split over several sections, as follows.

Section 3 is devoted to a single approach, Abstract Dialectical Frameworks (see for instance [21]), a general formalism for representing complex dependence links between arguments. In our terminology, the input consists of a dependence graph (nodes represent arguments and edges represent dependence links) together with an acceptance condition (in the form of a logical formula) attached to each node and the outputs are the logical models of the acceptance conditions, permitting to retrieve labellings and extensions.

The approaches reviewed in Section 4 aim at giving a logical theory (in propositional logic) that encodes an argumentation graph [30,33,52,67]. Multiple approaches are reviewed, that mainly vary on the logic used for the encoding: propositional logic, either pure or extended in a number of alternative ways

[1] For instance, an encoding of logic programs under the form of argumentation frameworks, or the creation of an argumentation framework using logical knowledge bases (see for instance, papers as [14]).

[2] Note that, even if we tried to give an uniform presentation of the different approaches, some of them will be sometimes over-represented, either because of the huge number of publications concerning them, or because of the need of some additional explanations (examples have been added in order to illustrate some complex notions used in the presented approaches).

(sorted language, modal-like language). Additional properties may map models to extensions according to a given semantics.

Section 5 reports on two approaches whose aim is to express properties over a given argumentation graph, so that these properties can be used to characterize appropriate labellings of the graph [4,49,50]. Thus, the input consists of an argumentation graph (or an abstract dialectical framework, see Section 3) together with an extension-based semantics σ. The main feature of both approaches is that the output is a Quantified Boolean Formula whose models coincide with the σ-extensions of the graph (i.e., using names of the nodes of the graph as propositional atoms, the models of the formula are exactly the σ-extensions of the graph).

Section 6 is mainly devoted to an approach called YALLA [55],[3] whose language permits not only to express an argumentation framework by means of specific formulae of first-order logic but also to express properties of update operators in dynamic argumentation. As an aside, a distinctive feature of YALLA is that a reference universe of argumentation is assumed, which makes it possible to capture cases of incomplete knowledge. The second approach reported in the same section proposes a propositional logic to specify and to check requirements in argumentation graphs [93]. The input consists of an argumentation graph together with constraints (such as: argument a or argument b is acceptable) and the outputs are formulae encoding the graph and the constraints, so that the models of the formulae capture properties of the argumentation graph.

In Section 7 are reviewed two approaches, [11,12] and [59], that, given an extension-based semantics σ, produce a formula Φ_σ whose satisfiability answers the σ-extension problem for the input (usually, an argumentation graph and a candidate subset of the arguments): given a candidate S, is S an extension according to the semantics σ?

The purpose of the two approaches reviewed in Section 8, [25,36], is to encode labelling-based semantics by means of a set of logical formulae (these express the different constraints associated with a particular kind of labelling). Thus, the input is an argumentation graph together with a labelling-based semantics s and the output is a logical theory characterizing the labelling-based semantics s (depending on s, it can be that the logic needed is second-order).

The purpose of the works reviewed in Section 9 is to associate a logic program with an argumentation graph in such a way that logic programming semantics, applied to the logic program, capture argumentative semantics. In this research line, many works exist from the seminal work presented by Dung [54]: for instance, [24,26,61,83,84,89]. So, this section describes different mappings which allow to transform an argumentation graph into a logic program, all of them offering different characterizations of argumentative semantics in terms of logic programming semantics.

Section 10 reviews three methods for expressing abstract argumentation in modal logic [25,70,93], two of them taking as input an argumentation graph together with a labelling (the third method regards argumentative semantics as primitives of the language) while the output consists of modal formulae expressing the distinctive properties of a given argumentative semantics.

Section 11 deals with approaches resorting to a constructive logic (either intuitionistic or Nelson's), where constructive negation is used to represent an attack in argumentation graphs and the models of the resulting formulae characterize the argumentative extensions of the input graph [63,68].

Lastly, Section 12 is an attempt to wrap all this up with proposing tentative conclusions suggested by the comparisons discussed throughout Sections 3–11.

[3] YALLA: Yet Another Logic Language for Argumentation.

Disclaimer. We have adopted (or at least attempted to) a single procedure to *present* all the works reviewed in our survey, in order to make it easier for the reader to compare these works. Please note that, in the last section, we go through the *same* example treated in turn by all approaches in order to better illustrate the behaviour of each of them. However, it is *not* our aim to assess them in any way, and we are definitely not to claim that such and such approach is better than another approach.

Moreover, despite the apparent similarity in the approaches (they are all based on logic and all attempt to capture Dung's argumentation model), it would be a formidable task to provide a comparative evaluation.

Finally, we make no claim for exhaustiveness. Some articles on the very topic of the survey may have gone unnoticed from us, others have been left out because we felt them having more emphasis on another topic or still other reasons. Of course, omitting to cite or to discuss these articles bears no quality assessment whatsoever on our behalf.

2. Abstract argumentation: Definitions and notations

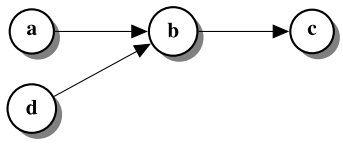
2.1. Argumentation frameworks

According to [54], an abstract argumentation framework consists of a set of arguments together with a binary relation between arguments.

Def. 1 (AF [54])**.** An *argumentation framework* (AF) is a pair $(\mathcal{A}, \mathcal{R})$ where $\mathcal{A}$ is a set[4] of abstract arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is a binary relation on $\mathcal{A}$, called the attack relation: $(a, b) \in \mathcal{R}$ means that a attacks b (a is the source of the interaction and b is the target).

An AF can be represented by a directed graph, called argumentation graph,[5] with vertices as arguments and edges as attacks.

Ex. 2.1.1. The AF defined by $\mathcal{A} = \{a, b, c, d\}$ and $\mathcal{R} = \{(a, b), (b, c), (d, b)\}$ can be represented by the following graph (arguments are given in a circle, and attacks are denoted by simple arrows):



Many extensions of this framework have been proposed. For instance, bipolar abstract frameworks have been introduced first in [71,91]. They include a second relation between arguments, the support relation, that is a positive interaction (in contrast to the attack relation that is a negative one). In [31], the support relation is left general so that the bipolar framework keeps a high level of abstraction.

Def. 2 (BAF [31])**.** A *bipolar argumentation framework* (BAF) is a triple $(\mathcal{A}, \mathcal{R}_{\text{att}}, \mathcal{R}_{\text{sup}})$ where $\mathcal{A}$ is a set of abstract arguments, $\mathcal{R}_{\text{att}} \subseteq \mathcal{A} \times \mathcal{A}$ (resp. $\mathcal{R}_{\text{sup}} \subseteq \mathcal{A} \times \mathcal{A}$) is a binary relation on $\mathcal{A}$, called the attack (resp. support) relation.

[4] Generally assumed to be finite.

[5] We will often use "argumentation graph" in place of "argumentation framework".

A BAF can still be represented by a directed graph with vertices as arguments and two kinds of edges (attacks denoted by simple arrows and supports denoted by double arrows).

Ex. 2.1.2. Consider the following BAF with only one support.

However, there is no single interpretation of the support, and a number of researchers proposed specialized variants of the support relation (deductive support [17], necessary support [79,80], evidential support [81,82]). These proposals have been developed quite independently, based on different intuitions and with different formalizations. In [32], is presented a comparative study in order to restate these proposals in a common setting, the bipolar argumentation framework (see also [41] for another survey).

For instance, evidential support is based on the intuition that every argument must be supported by some chain of supports rooted in special arguments called *prima-facie*. Considering a BAF with an evidential understanding of the support leads to Evidence-Based Argumentation Frameworks (EBAF) which can be defined as follows [29]:[6]

Def. 3 (EBAF)**.** An *Evidence-Based Argumentation Framework* (EBAF) is a 4-tuple $(\mathcal{A}, \mathcal{R}, \mathcal{E}, \mathcal{P})$ where $\mathcal{A}$ is a set of arguments, $\mathcal{R} \subseteq 2^{\mathcal{A}} \setminus \{\varnothing\} \times \mathcal{A}$ is the attack relation, $\mathcal{E} \subseteq 2^{\mathcal{A}} \setminus \{\varnothing\} \times \mathcal{A}$ is the support relation, and $\mathcal{P} \subseteq \mathcal{A}$ is the set of distinguished prima-facie arguments.

Another extension of AF is the higher-order AF with the idea of encompassing attacks to attacks in abstract argumentation frameworks (see [8] in the context of an extended framework handling argument strengths and their propagation). Then, higher-order attacks have been considered for representing preferences between arguments (second-order attacks in [76]), or for modelling situations where an attack might be defeated by an argument, without contesting the acceptability of the source of the attack [7]. Attacks to attacks and supports have been first considered in [64] with higher level networks, then in [92]; and more generally, in [42] an Attack–Support Argumentation Framework is proposed which allows for nested attacks and supports, i.e. attacks and supports whose targets can be other attacks or supports, at any level. Different names are given to these higher-order AF, depending on the kind of interaction that is handled: AFRA or RAF (with only attacks), ASAF or REBAF (with attacks and supports, necessary supports for ASAF and evidential supports for REBAF).

For instance, the definition of a RAF is as follows.

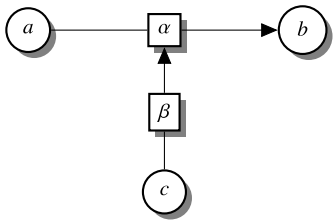
Def. 4 (RAF [28])**.** A *recursive argumentation framework (RAF)* is a tuple $\langle \mathcal{A}, \mathcal{R}, \mathbf{s}, \mathbf{t} \rangle$ where $\mathcal{A}$ is a finite and non-empty set of arguments, $\mathcal{R}$ is a finite set disjunct from $\mathcal{A}$ representing attack names, $\mathbf{s}$ is a function from $\mathcal{R}$ to $\mathcal{A}$ mapping each interaction to its source, and $\mathbf{t}$ is a function from $\mathcal{R}$ to $(\mathcal{A} \cup \mathcal{R})$ mapping each interaction to its target.

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

A RAF can also be represented graphically.

Ex. 2.1.3. The RAF in which an attack named α (with $\mathbf{s}(\alpha) = a$ and $\mathbf{t}(\alpha) = b \in \mathcal{A}$) being the target of an attack β (with $\mathbf{s}(\beta) = c$) can be represented by:

[6] The first definition of EBAF was given in [81] then modified in [87].



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

Still other extensions exist such as frameworks with collective interactions (SETAF: the source of interaction can be a set of arguments and not only one argument, as in EBAF or REBAF) and frameworks with weights over arguments or interactions.

2.2. *Acceptability semantics*

Case of AF. Acceptability semantics can be defined in terms of extensions [54] following basic requirements:

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

Standard AF semantics are defined as follows:

Def. 5 (Extension-based semantics [54])**.** Given $(\mathcal{A}, \mathcal{R})$ and $S \subseteq \mathcal{A}$.

- S is *conflict-free* iff $(a, b) \notin \mathcal{R}$ for all $a, b \in S$.
- $a \in \mathcal{A}$ is *acceptable* w.r.t. S (or equivalently S *defends* a) iff for each $b \in \mathcal{A}$ with $(b, a) \in \mathcal{R}$, there is $c \in S$ with $(c, b) \in \mathcal{R}$.
- The *characteristic function* of $(\mathcal{A}, \mathcal{R})$ is defined by: $\mathcal{F}(S) = \{a \in \mathcal{A}$ such that a is acceptable w.r.t. $S\}$.
- S is *admissible* iff S is conflict-free and $S \subseteq \mathcal{F}(S)$.
- S is *a complete extension* of $(\mathcal{A}, \mathcal{R})$ iff it is conflict-free and a fixed point of $\mathcal{F}$.
- S is *the grounded extension* of $(\mathcal{A}, \mathcal{R})$ iff it is the minimal (w.r.t. $\subseteq$) fixed point[7] of $\mathcal{F}$.
- S is *a preferred extension* of $(\mathcal{A}, \mathcal{R})$ iff it is a maximal (w.r.t. $\subseteq$) complete extension.
- S is *a stable extension* of $(\mathcal{A}, \mathcal{R})$ iff it is conflict-free and for each $a \notin S$, there is $b \in S$ with $(b, a) \in \mathcal{R}$.

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

Ex. 2.1.1 (cont'd) In this example, the set $\{a, c, d\}$ is the grounded, complete, preferred and stable extension.

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

Acceptability semantics can also be defined in terms of labellings, as in [6], for instance.

Def. 6 (Labelling [6])**.** Let $(\mathcal{A}, \mathcal{R})$ be an AF. A *labelling* for $(\mathcal{A}, \mathcal{R})$ is a total function $\ell : \mathcal{A} \rightarrow \{\texttt{in}, \texttt{out}, \texttt{und}\}$.

Let ℓ be a labelling of $(\mathcal{A}, \mathcal{R})$.

- An `in`-labelled argument is said to be *legally* `in` iff all its attackers are labelled `out`.
- An `out`-labelled argument is said to be *legally* `out` iff at least one of its attackers is labelled `in`.
- An `und`-labelled argument is said to be *legally* `und` iff it doesn't have an attacker that is labelled `in` and not all its attackers are labelled `out`.

Standard labelling-based semantics are defined as follows:

Def. 7 (Labelling-based semantics [6])**.** Let ℓ be a labelling for $(\mathcal{A}, \mathcal{R})$.

- ℓ is an *admissible labelling* iff each `in`-labelled argument is *legally* `in` and each `out`-labelled argument is *legally* `out`.
- ℓ is a *complete labelling* iff each `in`-labelled argument is *legally* `in`, each `out`-labelled argument is *legally* `out`, and each `und`-labelled argument is *legally* `und`.
- ℓ is the *grounded labelling* iff it is the complete labelling that minimizes (w.r.t. $\subseteq$) the set of `in`-labelled arguments.
- ℓ is a *preferred labelling* iff it is a complete labelling of $(\mathcal{A}, \mathcal{R})$ that maximizes (w.r.t. $\subseteq$) the set of `in`-labelled arguments.
- ℓ is a *stable labelling* iff it is a complete labelling with no `und`-labelled argument.

Alternative characterizations of complete labellings can be found in [25]. Let us recall them as they will be used in the remainder of this document:

Prop. 1 (Characterizing complete labellings)**.** *Let ℓ be a labelling for $(\mathcal{A}, \mathcal{R})$.*

(1) *ℓ is a* complete labelling *iff for each argument a, it holds that:*

- *If a is* `in`*-labelled, then all its attackers are* `out`*-labelled*
- *If a is* `out`*-labelled, then it has at least one attacker that is* `in`*-labelled*
- *If a is* `und`*-labelled, then it has at least one attacker that is not* `out`*-labelled and it does not have an attacker that is* `in`*-labelled*

(2) *ℓ is a* complete labelling *iff for each argument a, it holds that:*

- *a is* `in`*-labelled iff all its attackers are* `out`*-labelled*
- *a is* `out`*-labelled iff it has at least one attacker that is* `in`*-labelled.*

Case of extended AF (BAF, RAF). The associated semantics are very often defined using a flattening process: the extended AF is turned into an AF, then the AF semantics are applied (see for instance [7]). In recent works (see for instance [29]), semantics for extended AF have been defined directly.

3. ADF (Abstract Dialectical Frameworks)

The aim of the ADF approach is the definition of a general framework for representing complex dependence links between arguments (these links impact the acceptability of the arguments). It is rather

a theoretical approach but several implementations have been proposed. This approach has been introduced many years ago and one of the more recent synthetic paper about ADF is [21].

The input consists of a dependence graph (nodes are arguments, statements or positions and edges are dependence links) together with an acceptance condition attached to each node. This condition can be a propositional formula[8] expressing the way the status of the argument is impacted by the status of its parents in the graph. Following the acceptance conditions, one can encode an AF, a BAF (with different meanings for the support relation), a framework with sets of attacking arguments (SETAF). One can also encode an AF with higher-order interactions through the addition of meta-arguments representing interactions.

The outputs are the interpretations of the set of acceptance conditions. These interpretations allow to retrieve some labellings and extensions.

In this approach, the logic is used for encoding and interpretating the dependence links.

3.1. Main definitions

An ADF is a directed graph defined as follows:

Def. 8. An *abstract dialectical framework (ADF)* is a tuple $D = (S, L, C)$ where S is a finite set of nodes, $L \subseteq S \times S$ is a set of links, and $C = \{\varphi_s | s \in S\}$ is a set of propositional formulae (acceptance conditions). Each $\varphi_s \in C$ is built over the set $par(s) = \{a \in S | (a, s) \in L\}$ (i.e. $par(s)$ is the set of the parents of s).

So the status of an argument depends on the status of its parents in the graph following the acceptance condition attached to the argument. The ideas are the following ones:

> First, the status "accepted" (resp. "rejected", "unknown") of an argument is related to its assignment with the truth value t (resp. f, u).[9]
> Second, let s be an argument, let $par(s)$ be the set of the parents of s in the dependence graph, let φ_s be the acceptance condition associated with s and consider $R \subseteq par(s)$. $\varphi_s(R)$ denotes the truth value of φ_s using the truth value of the elements of R. Then $\varphi_s(R)$ is used for determining the status of s: if $\varphi_s(R) = t$ (resp. f, u) then s is accepted (resp. rejected, unknown).

Note that the dependence links can be extracted from the acceptance conditions. So in general the ADF is defined only with the set of arguments and their acceptance conditions: $D = (S, \{\varphi_s | s \in S\})$.

ADF semantics. In terms of semantics, two operators can be defined, one for two-values semantics and another one for three-valued semantics. The idea behind these operators is the following one: starting from a given interpretation, the use of the operator allows to browse the set of possible interpretations (two-valued or three-valued) taking into account the impact of acceptance conditions on the interpretations. The aim is to find a fixed point if it exists (it is always the case for three-valued operators).

Def. 9 (Two-valued operator)**.** Let $D = (S, \{\varphi_s | s \in S\})$ be an ADF. The two-valued operator G_D takes a two-valued interpretation v of each argument and returns a two-valued interpretation v' mapping each argument s to the truth value that is obtained by evaluating its acceptance condition φ_s with v.

Thus, for v a two-valued interpretation, $G_D(v) = v'$ where for $s \in S$, $v'(s) = v(\varphi_s)$.

[8] In ADF, this condition is generally a propositional formula, but there also exist some works on ADF that specify acceptance conditions as Boolean functions that may be represented in different ways. In this survey, we only consider the case where acceptance conditions are propositional formulae.

[9] t for *true*, f for *false* and u for *unknown*.

In the case of a three-valued operator, the idea is similar. However, due to the existence of the third value (u for unknown) a specific technique must be used:

Def. 10 (Three-valued operator)**.** Let $D = (S, \{\varphi_s | s \in S\})$ be an ADF. The three-valued operator Γ_D takes a three-valued interpretation v of each argument and returns a three-valued interpretation v' corresponding to the consensus truth value for the acceptance condition where this consensus takes into account all possible two-valued interpretations w that extend the input interpretation v.

Then semantics for interpretations can be defined using either a two-valued operator or a three-valued operator and a preordering between interpretations. Two preorderings can be defined:

Def. 11 (Preordering $\leqslant_t$ on two-valued interpretations)**.** Let $(S, \{\varphi_s | s \in S\})$ be an ADF. Let v_1 and v_2 be two-valued interpretations. $v_1 \leqslant_t v_2$ if and only if $\forall s \in S$, $v_1(s) = t \Rightarrow v_2(s) = t$.

Def. 12 (Preordering $\leqslant_i$ on three-valued interpretations)**.** Let $(S, \{\varphi_s | s \in S\})$ be an ADF. Let v_1 and v_2 be three-valued interpretations. $v_1 \leqslant_i v_2$ if and only if $\forall s \in S$, $v_1(s) \in \{t, f\} \Rightarrow v_2(s) = v_1(s)$.

Some semantics can be defined as follows:[10]

Def. 13. Let $D = (S, \{\varphi_s | s \in S\})$ be an ADF. Let v be a three-valued interpretation.

- v is *complete* for D iff $v = \Gamma_D(v)$.
- v is *admissible* for D iff $v \leqslant_i \Gamma_D(v)$.
- v is *preferred* for D iff v is $\leqslant_i$-maximal admissible.
- v is *grounded* for D iff v is the $\leqslant_i$-least fixed point of Γ_D.

Note that the well-known relationships between AF semantics also hold for ADF semantics:

Prop. 2. *Let $D = (S, \{\varphi_s | s \in S\})$ be an ADF.*

- *Each stable model of D is a two-valued model of D.*
- *Each two-valued model of D is a preferred interpretation of D.*
- *Each preferred interpretation of D is complete.*
- *Each complete interpretation of D is admissible for D.*
- *The grounded interpretation of D is complete.*[11]

ADF as a modelling tool. An ADF can be used for modelling variants of abstract argumentation frameworks such as AF, BAF, RAF. A significant survey can be found in [85,86].

- *Case of AF:* Let $(\mathcal{A}, \mathcal{R})$ be an AF, its associated ADF is defined by the pair $(\mathcal{A}, \{\varphi_a | a \in \mathcal{A}\})$ with $\varphi_a = \bigwedge_{b \in \mathcal{A}, (b,a) \in \mathcal{R}} \neg b$. Each Dung's semantics can be retrieved using ADF semantics.
- *Case of AF with annotated links and/or preferences:* An ADF can be used for expressing the impact of annotated links representing qualitative or quantitative preferences and/or preorderings between arguments.
  First, consider the case of a weighted AF (i.e. an AF with numerical weights attached to each interaction representing quantitative preferences).

[10] The definition for the stable models is not given here but can be found in [21] (a *model* being an interpretation v such that $\forall s \in S$, $v(s) = v(\varphi_s)$).

[11] Note that the grounded interpretation is unique.

* A positive (resp. negative) value on a link means that this link is a support (resp. attack) link;
* A link is said "active" if its source node is accepted;
* A node will be accepted if the sum of the weights of all the active links pointing to it is positive (strategy *sum-of-weights – sow*).

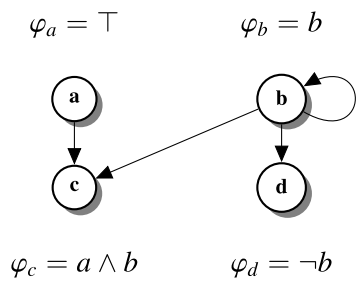
Example 3.2.6 given in the next section illustrates the above ideas.
Qualitative preference can be handled in a similar way, particularly for the input arguments. Note also that these kinds of annotated links can be used for expressing richer interactions such as, for instance, those used in legal reasoning (notions of "valid", "strong", "credible" and "weak" arguments and principles of "scintilla of evidence", "preponderance of evidence").

- *Case of BAF:* An annotated AF can be used for modelling a BAF (particularly the qualitative version with $+$ on the support links and $-$ on the attack links). Nevertheless, as different meanings exist for the support, several encodings can be defined (see Ex. 2.1.2 in the next section).
- *Case of AFRA, RAF:* In an ADF, no recursive link exists. So in order to represent an AFRA or a RAF with an ADF, flattening techniques must be used (for instance, the one described in [7] for translating an AFRA into an AF). See Ex. 2.1.3 in the next section.

3.2. Some examples

The previous ADF definitions are illustrated on the following examples (most of them are issued from [21]).

Ex. 3.2.1. Consider the ADF represented by:



Intuitively, φ_a states that a should be accepted. Condition φ_b expresses a kind of self-support for b. φ_c says that c will be accepted if both a and b are accepted whereas φ_d says that d is attacked by b. Note that strictly speaking, the attack from b to d is represented by the assertion "If b is accepted then d is not accepted" or equivalently "If d is accepted then b is not accepted", which is a necessary condition for the acceptance of d. It is worth noticed that such statements refer, implicitly, to argumentation semantics. The converse condition which writes "If b is not accepted then d is accepted" expresses a kind of reinstatement. So, at least for the attack links, the acceptance condition is a necessary and sufficient condition.

The mechanism used for the two-valued operator is illustrated by Fig. 3 (given at the end of this paper) that represents the evolution of interpretations by the operator G_D (nodes = interpretations and edges = the relation between two interpretations defined by the operator).

Consider for instance $v = \{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto t\}$. For each acceptance condition, it can be seen how the operator produces the updated interpretation v':

- $\varphi_a = \top$: so $v'(a) = t$ (v has no impact since a has no parent in the ADF);
- $\varphi_b = b$: so $v'(b) = v(b) = t$ (v has an impact since b is its own parent);
- $\varphi_c = a \wedge b$: so $v'(c) = v(a) \wedge v(b) = t$ (v has an impact since a and b are the parents of c);
- $\varphi_d = \neg b$: so $v'(d) = \neg v(b) = f$ (v has an impact since b is the parent of d).

So, the operator applied to $v = \{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto t\}$ produces the interpretation $G_D(v) = v' = \{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto f\}$.

Consider now $v' = \{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto f\}$ and, using the same way, compute the interpretation $G_D(v') = v''$. It is easy to see that v'' is exactly v'. So v' is a fixed point for the operator G_D. Note that, in this example, there exist 2 fixed points (each fixed point being by definition a two-valued model).

This example also illustrates the preordering $\leqslant_t$, see Fig. 4 (given at the end of this paper). Note that the set of two-valued interpretations over $\leqslant_t$ consists of a complete lattice (the top element of the lattice is at the top of the figure and the bottom element is at the bottom of the figure).

Unfortunately, due to the important number of three-valued interpretations of this example ($3^4 = 81$ three-valued interpretations), we do not represent the $\leqslant_i$ preordering and the corresponding complete meet-lattice in a figure. Nevertheless, some interesting three-valued interpretations can be identified:

- $v_0 = \{a \mapsto t, b \mapsto u, c \mapsto u, d \mapsto u\}$,
- $v_1 = \{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto f\}$ that is also a two-valued interpretation and a fixed point of G_D (so a two-valued model),
- $v_2 = \{a \mapsto t, b \mapsto f, c \mapsto f, d \mapsto t\}$ that is also a two-valued interpretation and a fixed point of G_D (so a two-valued model),
- $v = \{a \mapsto t, b \mapsto f, c \mapsto f, d \mapsto u\}$.

Following Def. 13, v_0, v_1 and v_2 are complete (they are the only fixed points of Γ_D). Moreover v_1 and v_2 are preferred and v_0 is grounded. It can be shown that v is admissible.[12]

The next example illustrates the three-valued operator and the $\leqslant_i$ preordering.

Ex. 3.2.2. Consider the ADF represented by:

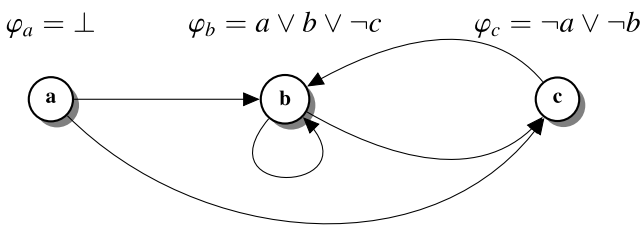


Figure 5 (given at the end of this paper) represents the evolution of interpretations by the three-valued operator Γ_D for the ADF (nodes = interpretations and edges = the relation between two interpretations defined by the operator).

First, consider $v = \{a \mapsto u, b \mapsto f, c \mapsto t\}$. The two possible two-valued interpretations that extend v are: $w_1 = \{a \mapsto t, b \mapsto f, c \mapsto t\}$ and $w_2 = \{a \mapsto f, b \mapsto f, c \mapsto t\}$. Consider now each acceptance condition and let us see how the operator produces the updated interpretation v':

[12] Moreover, v_2 is also the stable model of the ADF.

- With w_1:
  - $\varphi_a = \bot$: so $w'_1(a) = f$ (w_1 has no impact since a has no parent in the ADF);
  - $\varphi_b = a \vee b \vee \neg c$: so $w'_1(b) = t$ (w_1 has an impact since b as three parents – a, b and c –);
  - $\varphi_c = \neg a \vee \neg b$: so $w'_1(c) = t$ (w_1 has an impact since c as two parents – a and b –);
- With w_2:
  - $\varphi_a = \bot$: so $w'_2(a) = f$ (w_2 has no impact since a has no parent in the ADF);
  - $\varphi_b = a \vee b \vee \neg c$: so $w'_2(b) = f$ (w_2 has an impact since b as three parents – a, b and c –);
  - $\varphi_c = \neg a \vee \neg b$: so $w'_2(c) = t$ (w_2 has an impact since c as two parents – a and b –);

Then, using w'_1 and w'_2, one can compute the consensus truth value for each argument: $v'(a) = f$ (since $w'_1(a) = w'_2(a) = f$), $v'(b) = u$ (since $w'_1(b) \neq w'_2(b)$), $v'(c) = t$ (since $w'_1(c) = w'_2(c) = t$).

Another example of this process can be given with the interpretation $v' = \{a \mapsto f, b \mapsto u, c \mapsto t\}$. The two possible two-valued interpretations that extend v' are: $w'_1 = \{a \mapsto f, b \mapsto t, c \mapsto t\}$ and $w'_2 = \{a \mapsto f, b \mapsto f, c \mapsto t\}$.

With w'_1, we obtain $w''_1(a) = f$, $w''_1(b) = t$ and $w''_1(c) = t$.

With w'_2, we obtain $w''_2(a) = f$, $w''_2(b) = f$, $w''_2(c) = t$.

Then, using w''_1 and w''_2, the consensus truth value for each argument is $v''(a) = f$, $v''(b) = u$, $v''(c) = t$. This three-valued interpretation is one of the three fixed points of this example.

In this example, the preordering $\leqslant_i$ is given by Fig. 6 (given at the end of this paper). Note that the set of three-valued interpretations over $\leqslant_i$ consists of a complete meet-lattice (the top elements of the meet-lattice are at the top of the figure and the bottom element is at the bottom of the figure).

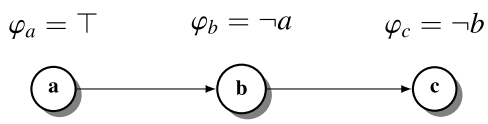
Here are some interesting three-valued interpretations:

- $v_0 = \{a \mapsto f, b \mapsto u, c \mapsto t\}$,
- $v_1 = \{a \mapsto f, b \mapsto t, c \mapsto t\}$ (that is also a two-valued model),
- $v_2 = \{a \mapsto f, b \mapsto f, c \mapsto t\}$ (that is also a two-valued model).

v_0, v_1 and v_2 are complete (they are the only fixed points of Γ_D), v_1 and v_2 are preferred and v_0 is grounded.

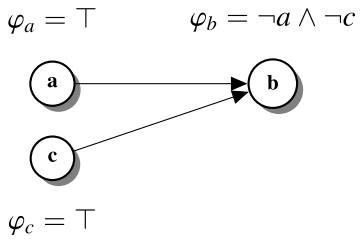
The following examples illustrate the use of ADF as a modelling tool.

Ex. 3.2.3. The sequence of two attacks can be translated into the following ADF:

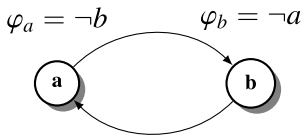


Applying the G_D or the Γ_D operators gives only one fixed point: $v = \{a \mapsto t, b \mapsto f, c \mapsto t\}$. That corresponds to the complete, preferred and grounded extension.

Ex. 3.2.4. Two attacks to the same argument can be translated into the following ADF:

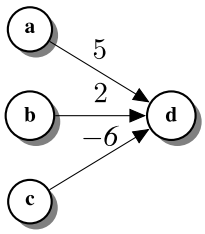


Ex. 3.2.5. An even-length cycle of attacks can be translated into the following ADF:

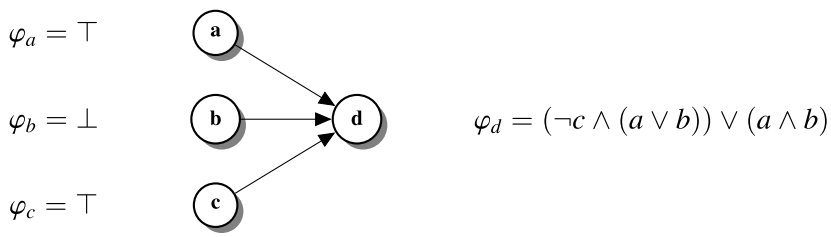


Applying the Γ_D operator gives three fixed points: $v_0 = \{a \mapsto u, b \mapsto u\}$, $v_1 = \{a \mapsto t, b \mapsto f\}$ and $v_2 = \{a \mapsto f, b \mapsto t\}$. That corresponds to the complete extensions, v_0 being the grounded one and v_1, v_2 being the preferred ones.

Ex. 3.2.6. The following graph represents a weighted AF.



Considering that the set $\{a, c\}$ is the subset of the input arguments that are accepted (so even if b is an input argument it is not considered as accepted), this weighted AF by an ADF can be turned into the following ADF:



Ex. 2.1.2 (cont'd) Consider a BAF with only one support from a to b. Following [21], the encoding of a support could be done using a positive link between a and b in the corresponding ADF. Nevertheless, in order to find the best encoding of the BAF into an ADS, we must identify the direction of this link and the acceptance conditions. These elements will depend on the meaning of the used support (indeed, recall that several meanings exist for the support relation in the literature); in this example, we are interested in the deductive and necessary supports. Moreover, in an ADF, it is important to note that the acceptance condition can be viewed as a sufficient and necessary condition.

Let consider the following possible encoding: a positive link from a to b with $\varphi_b = a$ and $\varphi_a = \top$. So, in this case, φ_b corresponds to the two following conditions: "if a is accepted then b is accepted" (that is the sufficient condition), and "a is accepted if b is accepted" (that is the necessary condition).

Consider now that the support is a deductive one. Then the meaning of this support in argumentation semantics can be described by the following assertion (the target of the support is impacted by its source):

"if a is accepted then b is accepted".

With the proposed encoding, the sufficient condition given by φ_b encodes exactly the deductive meaning of the support but the necessary condition issued from φ_b gives an additional constraint and we can easily see that this additional constraint corresponds to the necessary meaning of the support.

Indeed, the meaning of the necessary support from a to b in argumentation semantics can be described by the following assertion (the target of a support impacts its source):

"a is accepted if b is accepted".

The same problem appears when we try to represent a necessary support from a to b or when we use another encoding changing the direction of the link and the acceptance condition (a positive link from b to a with $\varphi_a = b$ and $\varphi_b = \top$). In each case, we obtain a formula that encodes both the deductive meaning of the support ("if a is accepted then b is accepted") and the necessary meaning of the support ("a is accepted if b is accepted"). Even if there is a difference between the two ADFs (the formula is attached either to b – deductive support –, or to a – necessary support –), when there is no other interaction from, or to, these two arguments, the acceptance conditions do not allow to distinguish between the two meanings of the supports in argumentation semantics.

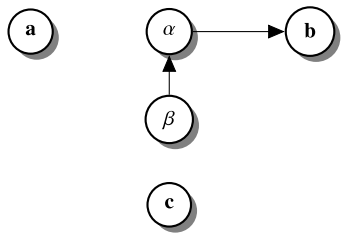
A synthesis of the above remarks is given in the following table. Let us consider a support from a to b, denoted by the pair (a, b):

in ADF	Sufficient condition of $\varphi_b = a$	Necessary condition of $\varphi_b = a$	Sufficient condition of $\varphi_a = b$	Necessary condition of $\varphi_a = b$
in BAF	Deductive meaning of (a, b)	Necessary meaning of (a, b)	Necessary meaning of (a, b)	Deductive meaning of (a, b)

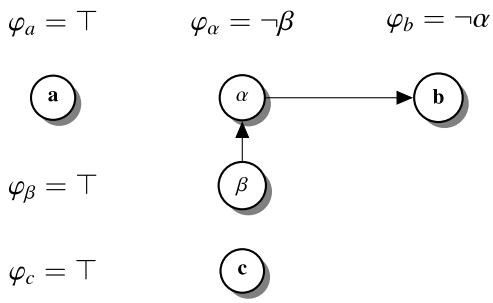
In conclusion, as the acceptance condition in an ADF reads as a necessary and sufficient condition, and without another mechanism (specific interpretations for instance), it seems difficult to capture a support which is not both necessary and deductive in argumentation semantics.

Ex. 2.1.3 (cont'd) Consider the RAF with an attack named α (from a to b) attacked by another attack named β whose source is c.

Using the flattening technique, this RAF can be turned into the following AF (the RAF is translated into an AF by the addition of new arguments, one for each attack, and the definition of a new set of attacks; for more details, see for instance [7]):



This AF can be in turn represented by the following ADF:



3.3. Some implementations

Several implementations have been proposed for ADF (see [22,23]). Here are some of them:[13]

DIAMOND (DIAlectictal MOdels eNcoDing) is based on Answer Set Programming (ASP). DIAMOND translates an ADF into an ASP program whose stable models correspond to models of the ADF with respect to several semantics (i.e. admissible, complete, stable, grounded).

QADF is a system for solving reasoning problems on ADF using Quantified Boolean Formulae (QBF).[14] Given an ADF and a reasoning problem as input, QADF returns the encoding of the reasoning problem as a QBF for that ADF. Then, a subsequent QBF solver solves the reasoning task.

UNREAL (Uniform Account of Realizability in Abstract Argumentation) is a system based on ASP for deciding realizability of a given set of interpretations. It supports AFs, ADFs, the subclass of bipolar ADFs (BADFs), and frameworks with sets of attacking arguments (SETAFs). For each of these formalisms, realizability can be tested for the standard semantics, namely admissible, complete, preferred and two-valued models (stable semantics for (SET)AFs). In words, given a set of interpretations V, a formalism F and a semantics σ, UNREAL computes all knowledge bases K of type F having $\sigma(K) = V$. Optionally, the output can be converted into the format readable by ASPARTIX (see Section 9.3) or DIAMOND.

GrappaVis is a Java tool for specifying and evaluating GRAPPA and ADF instances (GRAPPA being a semantical framework for graph-based argument processing − GRAPPA = GRaph-based Argument Processing based on Patterns of Acceptance). GrappaVis is a graphical tool for specifying GRAPPA and ADF-instances, evaluating them and visualizing the results of the evaluation.

[13] This list is not exhaustive. For instance, the reader can also refer to YADF [19] or to k++ADF [75].

[14] These formulae are a generalization of the propositional formulae in which both existential quantifiers and universal quantifiers can be applied to each variable.

GrappaVis itself is a JAVA-application based on the JGraphX framework and therefore provides intuitive tools to draw GRAPPA / ADF instances. For the evaluation it makes use of two different types of ASP encodings.

3.4. Related works

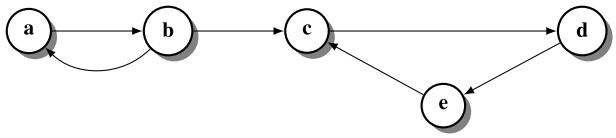
As shown before, the ADF approach is able to encompass several approaches of abstract argumentation. The main difficulty rests in the choice of the right acceptance conditions appropriate to a given framework.

The idea to attach an acceptance condition to an AF has also been used in [43]. In this approach, a constrained argumentation framework (CAF) has been defined as an AF associated with an acceptance condition that is a propositional formula. As in ADF, this propositional formula is defined over a vocabulary built on the set of arguments. Nevertheless there exist important differences between a CAF and an ADF:

- the graph used in [43] is a standard argumentation graph (links encode attacks) and not a dependence graph,
- there is a unique acceptance condition in [43] and not one attached to each argument,
- In [43] are defined new semantics by combining standard semantics with the satisfaction of the acceptance condition.

The following example illustrates this approach.

Ex. 3.4.1. Consider the CAF built with the following AF and the propositional formula $C = a \wedge \neg e$:



There are 2 preferred extensions $\{a\}$ and $\{b, d\}$ for the AF.

Then taking into account the acceptance condition C, only the first one can be considered as an extension for the CAF.

4. Translation of AF into propositional logic

The approaches reviewed in this section aim at giving a logical theory (in propositional logic) that encodes an AF. That is, for each of these approaches, the input is an AF and the output is a theory in propositional logic. They are theoretical approaches, but the output theories supplemented with formulae capturing a given argumentation semantics, can be fed into a SAT solver. In these approaches, the logic (propositional logic, either pure or extended in a number of alternative ways: sorted language, modal-like language) is used to encode the AF. Additional properties may map models to extensions according to a given semantics.

4.1. The syntactic sugar way

The simplest approach for encoding an AF consists of introducing special propositional atoms in the language in order to denote the edges of the argumentation graph ($\mathtt{Att}_{a,b}$ in [52], $r_{a,b}$ in [56][15]). These special atoms are considered as always *true*.

The description needs not include propositional atoms of the form $\mathtt{Arg}_a$ (or the like) that would explicitly list the nodes of the argumentation graph.

4.2. The approach by Cayrol et al.

This approach has been presented in several papers [30,33,34]. The definitions presented here are those given in the more recent paper [33].

In the case of an AF, the atoms are of the form $Acc(a)$ and $Nacc(a)$. For a node a of the argumentation graph, $Acc(a)$ expresses the status of being accepted, whereas $Nacc(a)$ expresses that a cannot be accepted (implicitly: with regard to a given argumentation semantics). Note that the meaning of $Nacc(a)$ is stronger than "a is not accepted".

The theory generated by [33] consists of the following axioms:

For every $a\mathcal{R}b$ in the input AF

$$\big(Acc(a) \rightarrow Nacc(b)\big)$$

For every argument b in the input AF

$$\big(Nacc(b) \rightarrow \neg Acc(b)\big)$$

Such formulae express (by transitivity of material implication) conflict-freeness of extension-based semantics: $Acc(a) \rightarrow \neg Acc(b)$ whenever $a\mathcal{R}b$. Such formulae also express that if an argument cannot be accepted then it is not accepted: $Nacc(b) \rightarrow \neg Acc(b)$, or, equivalently, $\neg(Acc(b) \wedge Nacc(b))$.

In the case of RAFs, the authors use a two-sorted logic with equality. There are $\mathtt{arg}$ a sort for arguments and $\mathtt{att}$ a sort for attacks. In addition to $Acc(a)$ and $Nacc(a)$ as above, the language also admits atoms of the form $Val(e)$ for attack names in the input RAF (intuitively, $Val(e)$ means that the attack named e is valid in the input RAF with regard to a given argumentation semantics). Lastly, two function symbols s and t can be applied to objects of the sort $\mathtt{att}$ to capture source and target of the attack. The target can be either of sort $\mathtt{arg}$ or of sort $\mathtt{att}$ and the source can only be of sort $\mathtt{arg}$ (as the source of an attack in a RAF is restricted to a single argument, see Definition 4, in Section 2).

The generated theory is bigger than for AFs. First, there are the following conflict-freeness axioms:

$$\forall e : \mathtt{att}, \quad \big(\big(Val(e) \wedge Acc\big(s(e)\big)\big) \rightarrow \neg Val\big(t(e)\big)\big) \quad \text{if } t(e) \text{ is of sort } \mathtt{att}$$

$$\forall e : \mathtt{att}, \quad \big(\big(Val(e) \wedge Acc\big(s(e)\big)\big) \rightarrow Nacc\big(t(e)\big)\big) \quad \text{if } t(e) \text{ is of sort } \mathtt{arg}$$

and the coherence axiom as above:

$$\forall a : \mathtt{arg}, \quad Nacc(a) \rightarrow \neg Acc(a)$$

[15] See also Section 6. In the YALLA language, a specific predicate has been introduced for encoding an attack between sets of arguments: $x \rhd y$.

Of course, the generated theory includes RAF-dependent axioms (assuming the arguments of the input RAF are $a_1, \ldots, a_n$ and the attack names are $e_1, \ldots, e_m$) as follows:

$$s(e) = x \wedge t(e) = y \quad \text{for every edge } e \text{ from } x \text{ to } y$$

$$\forall a : \texttt{arg}, \quad a = a_1 \vee \cdots \vee a = a_n$$

$$\forall e : \texttt{att}, \quad e = e_1 \vee \cdots \vee e = e_m$$

$$\neg(a_i = a_j) \wedge \neg(e_i = e_j) \quad i \neq j$$

Additional formulae are introduced for encoding the different principles that govern argumentation semantics. There are formulae for capturing the defence principle, the reinstatement principle and the stability principle. Then extensions under a given semantics (admissible, complete, preferred, grounded, or stable semantics) can be characterized by models of logical theories obtained by combining some of these formulae. See an example in Section 12.2.

4.3. The approach by Gabbay and Gabbay

In [67], for every AF, a theory is generated with the axiom:

$$Na \rightarrow \neg a, \quad \text{for all } a \in \mathcal{A},$$

where Na is a special propositional atom with the same reading as $Nacc(a)$ in the Cayrol–Fariñas–Lagasquie approach. Also, a is to play here the same role as $Acc(a)$ in the Cayrol–Fariñas–Lagasquie approach.

Similarly, a conflict-freeness axiom is needed:

$$a \rightarrow Nb, \quad \text{for all } a \text{ and } b \text{ such that } a\mathcal{R}b.$$

The theory generated by [67] consists of four kinds of formulae:

(1) $\{x | \neg\exists y(y\mathcal{R}x)\}$
(2) $\{y \leftrightarrow \bigwedge_{z\in\mathcal{R}^-(y)} Nz\}$ for each argument y[16]
(3) $\{z \rightarrow Ny | (z\mathcal{R}y)\}$
(4) $\{(\bigwedge_{z\in\mathcal{R}^-(y)} \neg z \wedge \bigvee_{z\in\mathcal{R}^-(y)} \neg Nz) \rightarrow \neg y \wedge \neg Ny\}$ for each argument y[17]

The complete extensions correspond to the models of the above theory. See an example in Section 12.2.

Note that the above formulae fail to provide a modular encoding of some principles of argumentation semantics such as for instance the defence or the reinstatement principles (so, it is not so easy to characterize, e.g., admissibility). This is a difference with the Cayrol–Fariñas–Lagasquie approach.

An additional axiom is introduced for characterizing stable semantics:

$$a \vee Na, \quad \text{for all } a.$$

[16] If y is unattacked then this formula is equivalent to $y \leftrightarrow \top$ as the conjunction of an empty set of formulae is always *true*.

[17] If y is unattacked then the formula $\bigvee_{z\in\mathcal{R}^-(y)} \neg Nz$ is equivalent to $\perp$ as the disjunction of an empty set of formulae is always *false*.

The corresponding stability axiom given in [33] is:

$$\neg Acc(a) \rightarrow \bigvee_{b \in \mathcal{R}^-(a)} Acc(b)$$

Both [30] and [67] prove that the models of the respective theory coincide with the extensions of the AF (the list of semantics dealt with in [30] is longer).

4.4. Some implementations

The approach of [30,33,34] has been implemented in GRAFIX, a Java software for representing graphically argumentation frameworks with higher-order interactions (see [35]).

5. QBF-formalization of AF

The aim of both approaches [4,50] reviewed in this section is to express properties over a given AF, so that these properties can be used to characterize extensions of the AF. Thus, the input consists of an AF (or an ADF [49]) together with an extension-based semantics. The output is a quantified Boolean formula (QBF)[18] whose models coincide with the extensions of the AF (i.e., using names of the nodes of the argumentation graph as propositional atoms, the models of the formula are exactly the extensions of the AF). The definitions are formulated in terms of labellings over the AF, and translated in a logical language (QBF). The translation is shown to establish a one-to-one correspondence (i.e., a bijection) between the extensions of the AF and the models of the resulting quantified Boolean formula. The logic is used to encode the AF (nodes of the argumentation graph are injectively mapped to propositional atoms and edges are injectively mapped to special propositional atoms that are taken to be always *true*), with quantification serving to express minimality/maximality w.r.t. a property (e.g., there exists no complete labelling larger than the current labelling). The propositional atoms denoting nodes are duplicated, with a plus version and a minus version (e.g., for a node of name a, the language at hand includes $a^{\oplus}$ and $a^{\ominus}$). These signed atoms are used to encode the truth values t, f, u according to the labels as assigned (e.g., if a is assigned u by the current labelling then this is captured by letting both $a^{\oplus}$ and $a^{\ominus}$ to be *false*). A general constraint rules out the possibility for $a^{\oplus}$ and $a^{\ominus}$ to be both *true*.

These are theoretical approaches but implementations are afoot using QBF solvers, and indeed such an implementation is QADF, due to Diller, Wallner & Woltran [50].

5.1. Main definitions

We first describe the approach due to Arieli and Caminada [4,5].

Let us recall that a labelling ℓ for an AF $(\mathcal{A}, \mathcal{R})$, is a total function $\ell\colon \mathcal{A} \rightarrow \{\mathtt{in}, \mathtt{out}, \mathtt{und}\}$. We consider complete labellings, characterized by the following conditions (see Section 2.2):

For each argument a,

- If a is `in`-labelled, then all its attackers are `out`-labelled;

[18] These formulae are a generalization of the propositional formulae in which both existential quantifiers and universal quantifiers can be applied to each variable. With this formalism, one can ask whether a quantified sentential form over a set of Boolean variables is *true* or *false*. An example of QBF formula: $\forall x \exists y \exists z((x \vee z) \wedge y)$.

- If a is `out`-labelled, then it has at least one attacker that is `in`-labelled;
- If a is `und`-labelled, then it has at least one attacker that is not `out`-labelled and it does not have an attacker that is `in`-labelled.

The assignments of truth values to (name of) nodes are captured syntactically, that is, through formulae. This is done as follows:

Def. 14. For an unsigned atom a and unsigned formulae ϕ, ψ, define the following signed formulae:

$$\tau^+(a) = a^\oplus \qquad \tau^-(a) = a^\ominus$$
$$\tau^+(\neg\phi) = \tau^-(\phi) \qquad \tau^-(\neg\phi) = \tau^+(\phi)$$
$$\tau^+(\phi \to \psi) = \neg\tau^-(\phi) \vee \tau^+(\psi) \qquad \tau^-(\phi \to \psi) = \tau^+(\phi) \wedge \tau^-(\psi)$$

The encoding then expands to

Def. 15. For an unsigned formula ϕ:

$$\mathtt{val}(\phi, t) = \tau^+(\phi) \wedge \neg\tau^-(\phi)$$
$$\mathtt{val}(\phi, f) = \neg\tau^+(\phi) \wedge \tau^-(\phi)$$
$$\mathtt{val}(\phi, u) = \neg\tau^+(\phi) \wedge \neg\tau^-(\phi)$$

The main definition is:[19]

Def. 16. Let $AF = (\mathcal{A}, \mathcal{R})$ and $x \in \mathcal{A}$. Define

$$LAB_{AF}(x) = \left\{ \begin{array}{c} \mathtt{val}(x,t) \to \bigwedge_{y\in\mathcal{A}}(\mathtt{att}(y,x) \to \mathtt{val}(y,f)) \\ \mathtt{val}(x,f) \to \bigvee_{y\in\mathcal{A}}(\mathtt{att}(y,x) \wedge \mathtt{val}(y,t)) \\ \mathtt{val}(x,u) \to \left(\begin{array}{c} (\neg \bigwedge_{y\in\mathcal{A}}(\mathtt{att}(y,x) \to \mathtt{val}(y,f))) \\ \wedge(\neg \bigvee_{y\in\mathcal{A}}(\mathtt{att}(y,x) \wedge \mathtt{val}(y,t))) \end{array} \right) \end{array} \right\}$$

The role of $LAB_{AF}(x)$ is to ensure that the three conditions for a complete labelling are satisfied, for x ranging over all arguments of AF. Then, the formula characterizing complete labellings over $AF = (\mathcal{A}, \mathcal{R})$ is:

$$CMP(AF) = \bigcup_{a\in\mathcal{A}} LAB_{AF}(a) \cup \left\{\neg\left(a^\oplus \wedge a^\ominus\right)\right\}$$

Also, stable extensions can be captured:

$$STB(AF) = CMP(AF) \cup \left\{a^\oplus \vee a^\ominus | a \in \mathcal{A}\right\}$$

[19] $\mathtt{att}(y, x)$ means that $(y, x) \in \mathcal{R}$.

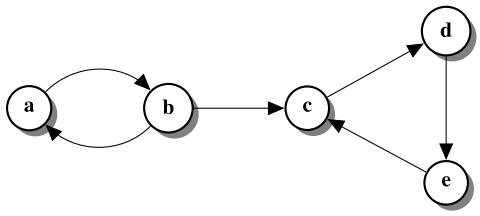
Quantified Boolean formulae enter the picture as minimization or maximization is required. For an AF such that $\mathcal{A} = \{a_1, \ldots, a_n\}$, an example is:

$$\forall x_1^{\oplus} x_1^{\ominus} \cdots x_n^{\oplus} x_n^{\ominus}$$
$$\left[CMP(AF)[x_1, \ldots, x_n] \rightarrow \begin{pmatrix} (\bigwedge_{a_i \in \mathcal{A}}(\mathtt{val}(x_i, t) \rightarrow \mathtt{val}(a_i, t))) \\ \rightarrow (\bigwedge_{a_i \in \mathcal{A}}(\mathtt{val}(a_i, t) \rightarrow \mathtt{val}(x_i, t))) \end{pmatrix} \right]$$

This formula can be read as follows. Considering a current labelling $a_1, \ldots, a_n$ for the n arguments in $\mathcal{A}$, look at all labellings $x_1, \ldots, x_n$ so that if one of them satisfies $CMP(AF)$ and is smaller than the current labelling (this is what $\bigwedge_{a_i \in \mathcal{A}}(\mathtt{val}(x_i, t) \rightarrow \mathtt{val}(a_i, t))$ expresses) then the current labelling must be in fact the same as this labelling. In other words, any model of this formula and of $CMP(AF)[a_1, \ldots, a_n]$ defines a minimal complete extension, so that (according to well-known results in abstract argumentation) it defines the grounded extension.

5.2. An example

Ex. 3.4.1 (cont'd)



The corresponding theory is:[20]

$$\mathtt{val}(a, t) \rightarrow \mathtt{val}(b, f)$$
$$\mathtt{val}(b, t) \rightarrow \mathtt{val}(a, f)$$
$$\mathtt{val}(c, t) \rightarrow \big(\mathtt{val}(b, f) \wedge \mathtt{val}(e, f)\big)$$
$$\mathtt{val}(d, t) \rightarrow \mathtt{val}(c, f)$$
$$\mathtt{val}(e, t) \rightarrow \mathtt{val}(d, f)$$

$$\mathtt{val}(a, f) \rightarrow \mathtt{val}(b, t)$$
$$\mathtt{val}(b, f) \rightarrow \mathtt{val}(a, t)$$
$$\mathtt{val}(c, f) \rightarrow \big(\mathtt{val}(b, t) \vee \mathtt{val}(e, t)\big)$$
$$\mathtt{val}(d, f) \rightarrow \mathtt{val}(c, t)$$
$$\mathtt{val}(e, f) \rightarrow \mathtt{val}(d, t)$$

[20] This set of rules is partitioned in 3 subsets: rules for assigning to each argument the value t, then f, then u.

$$\mathtt{val}(a,u) \rightarrow \big(\neg\mathtt{val}(b,f) \wedge \neg\mathtt{val}(b,t)\big)$$
$$\mathtt{val}(b,u) \rightarrow \big(\neg\mathtt{val}(a,f) \wedge \neg\mathtt{val}(a,t)\big)$$
$$\mathtt{val}(c,u) \rightarrow \big(\neg\big(\mathtt{val}(b,f) \wedge \mathtt{val}(e,f)\big) \wedge \neg\big(\mathtt{val}(b,t) \vee \mathtt{val}(e,t)\big)\big)$$
$$\mathtt{val}(d,u) \rightarrow \big(\neg\mathtt{val}(c,f) \wedge \neg\mathtt{val}(c,t)\big)$$
$$\mathtt{val}(e,u) \rightarrow \big(\neg\mathtt{val}(d,f) \wedge \neg\mathtt{val}(d,t)\big)$$

In the signed language,

$$\big(a^{\oplus} \wedge \neg a^{\ominus}\big) \rightarrow \big(b^{\ominus} \wedge \neg b^{\oplus}\big)$$
$$\big(b^{\oplus} \wedge \neg b^{\ominus}\big) \rightarrow \big(a^{\ominus} \wedge \neg a^{\oplus}\big)$$
$$\big(c^{\oplus} \wedge \neg c^{\ominus}\big) \rightarrow \big(\big(b^{\ominus} \wedge \neg b^{\oplus}\big) \wedge \big(e^{\ominus} \wedge \neg e^{\oplus}\big)\big)$$
$$\big(d^{\oplus} \wedge \neg d^{\ominus}\big) \rightarrow \big(c^{\ominus} \wedge \neg c^{\oplus}\big)$$
$$\big(e^{\oplus} \wedge \neg e^{\ominus}\big) \rightarrow \big(d^{\ominus} \wedge \neg d^{\oplus}\big)$$

$$\big(a^{\ominus} \wedge \neg a^{\oplus}\big) \rightarrow \big(b^{\oplus} \wedge \neg b^{\ominus}\big)$$
$$\big(b^{\ominus} \wedge \neg b^{\oplus}\big) \rightarrow \big(a^{\oplus} \wedge \neg a^{\ominus}\big)$$
$$\big(c^{\ominus} \wedge \neg c^{\oplus}\big) \rightarrow \big(\big(b^{\oplus} \wedge \neg b^{\ominus}\big) \vee \big(e^{\oplus} \wedge \neg e^{\ominus}\big)\big)$$
$$\big(d^{\ominus} \wedge \neg d^{\oplus}\big) \rightarrow \big(c^{\oplus} \wedge \neg c^{\ominus}\big)$$
$$\big(e^{\ominus} \wedge \neg e^{\oplus}\big) \rightarrow \big(d^{\oplus} \wedge \neg d^{\ominus}\big)$$

$$\big(\neg a^{\oplus} \wedge \neg a^{\ominus}\big) \rightarrow \big(\neg\big(b^{\ominus} \wedge \neg b^{\oplus}\big) \wedge \neg\big(b^{\oplus} \wedge \neg b^{\ominus}\big)\big)$$
$$\big(\neg b^{\oplus} \wedge \neg b^{\ominus}\big) \rightarrow \big(\neg\big(a^{\ominus} \wedge \neg a^{\oplus}\big) \wedge \neg\big(a^{\oplus} \wedge \neg a^{\ominus}\big)\big)$$
$$\big(\big(\neg c^{\oplus} \wedge \neg c^{\ominus}\big) \rightarrow \big[\neg\big(\big(b^{\ominus} \wedge \neg b^{\oplus}\big) \wedge \big(e^{\ominus} \wedge \neg e^{\oplus}\big)\big) \wedge \neg\big(\big(b^{\oplus} \wedge \neg b^{\ominus}\big) \vee \big(e^{\oplus} \wedge \neg e^{\ominus}\big)\big)\big]$$
$$\neg d^{\oplus} \wedge \neg d^{\ominus}\big) \rightarrow \big(\neg\big(c^{\ominus} \wedge \neg c^{\oplus}\big) \wedge \neg\big(c^{\oplus} \wedge \neg c^{\ominus}\big)\big)$$
$$\big(\neg e^{\oplus} \wedge \neg e^{\ominus}\big) \rightarrow \big(\neg\big(d^{\ominus} \wedge \neg d^{\oplus}\big) \wedge \neg\big(d^{\oplus} \wedge \neg d^{\ominus}\big)\big)$$

$$\neg\big(a^{\oplus} \wedge a^{\ominus}\big) \wedge \neg\big(b^{\oplus} \wedge b^{\ominus}\big) \wedge \neg\big(c^{\oplus} \wedge c^{\ominus}\big) \wedge \neg\big(d^{\oplus} \wedge d^{\ominus}\big) \wedge \neg\big(e^{\oplus} \wedge e^{\ominus}\big)$$

5.3. A QBF approach for ADF

The QBF-based formalization due to Diller, Wallner and Woltran [49,50] concerns ADFs.

This approach also uses a signed language, with the same atoms as given in Def. 14, namely $a^{\oplus}$ and $a^{\ominus}$ for every $a \in \mathcal{A}$ (interestingly, the notation for these atoms is the same in both approaches).

The same constraint (see *CMP*(*AF*) above) ruling out a contradictory assignment of truth values is enforced, i.e., for every $a \in \mathcal{A}$:

$$\neg\left(a^{\oplus} \wedge a^{\ominus}\right)$$

The main difference with the approach due to Arieli and Caminada is that the existence of the acceptance condition φ_a in ADF is addressed by Boolean quantification, even prior to any minimization/maximization:[21]

$$\forall x_1^{\oplus} x_1^{\ominus} \cdots x_n^{\oplus} x_n^{\ominus}$$
$$\left[\bigwedge_{a \in \mathcal{A}} \left(\left(a^{\oplus} \rightarrow a\right) \wedge \left(a^{\ominus} \rightarrow \neg a\right)\right) \rightarrow \bigwedge_{a \in \mathcal{A}} \left(\left(a^{\oplus} \rightarrow \varphi_a\right) \wedge \left(a^{\ominus} \rightarrow \neg \varphi_a\right)\right)\right]$$

This formula expresses that the current labelling is admissible. It can then be used to obtain a formula *CMP* for complete labellings, again using Boolean quantification. In turn, Boolean quantification over *CMP* can be used to obtain, e.g., a formula characterizing the grounded extension.

5.4. Implementation

The QBF approach has been implemented in the form of the QADF system (see the description in Section 3.3).

6. Dedicated languages for abstract argumentation

6.1. YALLA (Yet Another Logic Language for Argumentation)

The aim of this approach is the definition of a first-order logical theory capable of describing an AF and its standard semantics. In the basic language YALLA, an AF is described by specific axioms of the theory and formulae are interpreted by argumentation graphs. A variant of the basic language called $\text{YALLA}_{\mathbf{U}}$ has been defined for describing AFs built on a given *universe* U. Such a universe is supposed to specify exactly what arguments and interactions are possible w.r.t. the studied case.

It is a theoretical approach (no implementation yet) introduced in order to express the properties of update operators in dynamic argumentation. Moreover, $\text{YALLA}_{\mathbf{U}}$ enables to express incomplete knowledge about an AF, and to describe a set of AFs by one formula (each model of this formula corresponding to a particular AF).

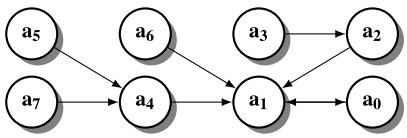
6.1.1. Main definitions

[55] has proposed a framework for handling change in argumentation (addition or removal of arguments or attacks). All the definitions are related to a specific AF, called *universe*, setting the set of possible arguments together with their interactions. This universe is supposed to be finite and is denoted by the pair $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$. For instance, if the domain is a knowledge base then $\mathcal{A}_{\mathbf{U}}$ and $\mathcal{R}_{\mathbf{U}}$ are the set of all arguments and interactions that may be built from the formulae of the base. In the following example issued from [55], it is assumed that $\mathcal{A}_{\mathbf{U}}$ and $\mathcal{R}_{\mathbf{U}}$ are explicitly provided:

[21] The quantification ensures ranging over all interpretations: conjunction over all $a \in \mathcal{A}$ fails to bind other propositional atoms that presumably occur in φ_a.

Ex. 6.1.1. During a trial concerning a defendant (Mr. X), several arguments can be involved to determine his guilt. The set of arguments $\mathcal{A}_{\mathbf{U}}$ and the graphical representation of the relation $\mathcal{R}_{\mathbf{U}}$ are given below.

a_0	*Mr. X is not guilty of premeditated murder of Mrs. X, his wife.*
a_1	*Mr. X is guilty of premeditated murder of Mrs. X.*
a_2	*The defendant has an alibi, his business associate has solemnly sworn that he met him at the time of the murder.*
a_3	*The close working business relationships between Mr. X and his associate induce suspicions about his testimony.*
a_4	*Mr. X loves his wife so deeply that he asked her to marry him twice. A man who loves his wife cannot be her killer.*
a_5	*Mr. X has a reputation for being promiscuous.*
a_6	*The defendant had no interest to kill his wife, since he was not the beneficiary of the huge life insurance she contracted.*
a_7	*The defendant is a man known to be venal and his "love" for a very rich woman could be only lure of profit.*



In [55], an AF is defined w.r.t. to a given universe $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$, so the definition differs slightly from the definition of [54] in the sense that arguments and interactions must be built according to the universe.

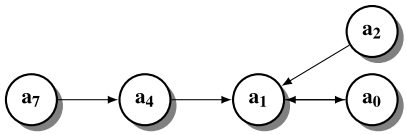
Def. 17. An AF on $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$ is a pair $(\mathcal{A}, \mathcal{R})$ where

- $\mathcal{A} \subseteq \mathcal{A}_{\mathbf{U}}$ and
- $\mathcal{R} \subseteq \mathcal{R}_{\mathbf{U}} \cap (\mathcal{A} \times \mathcal{A})$.

The set of AFs that can be built on the universe $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$ is denoted by $\Gamma_{\mathbf{U}}$.

An example of AF on the universe $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$ described in Ex. 6.1.1 could be:

Ex. 6.1.2. The prosecutor is trying to make accepted the guilt of Mr. X. She is not omniscient and knows only a subset of the arguments of the universe presented in Example 6.1.1 (a subset that is not necessarily shared with other agents). Moreover, her knowledge being based on the universe, any argument or attack that does not appear in the universe cannot appear in her graph. Here is her AF (AF_{Pro}):



[55] has proposed a first-order logical theory capable of describing abstract AFs built on a given *finite* universe $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$, where $\mathcal{A}_{\mathbf{U}} = \{a_1, a_2, \ldots, a_k\}$ with k being the cardinal of $\mathcal{A}_{\mathbf{U}}$. The signature of the associated language YALLA$_{\mathbf{U}}$ is defined as follows:[22]

Def. 18 (Signature). $\Sigma_{\mathbf{U}} = (V_{\text{const}}, V_f, V_P)$ where the set of constants $V_{\text{const}} = \{c_\perp, c_1, \ldots, c_p\}$ with $p = 2^k - 1$, the set of functions $V_f = \{union^2\}$ and the set of predicates $V_P = \{on^1, \rhd^2, \subseteq^2\}$.

[22] For each function or predicate symbol, the arity is indicated by an exponent attached to the symbol.

As the logical theory has been built for describing AFs on a given universe, terms and formulae of the language $\text{YALLA}_{\mathbf{U}}$ will be interpreted on AFs built on this universe. Formally, the semantics of $\text{YALLA}_{\mathbf{U}}$ is defined thanks to a structure over $\Sigma_{\mathbf{U}}$, on which terms and formulae will be interpreted. So a structure is associated with an AF built on the universe $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$ and its domain is $\mathcal{D} = 2^{\mathcal{A}_{\mathbf{U}}}$, which is not empty.

Def. 19 (**Structure**). A *structure* $\mathcal{M}$ over the signature $\Sigma_{\mathbf{U}}$, associated with $(\mathcal{A}, \mathcal{R})$, is a pair $(\mathcal{D}, \mathcal{I})$ where $\mathcal{D} = 2^{\mathcal{A}_{\mathbf{U}}}$ is the *domain of the structure* and $\mathcal{I}$ is an *interpretation function* associating:

(1) a unique element of $\mathcal{D}$ to each constant symbol c_i (in particular the empty set is associated with the constant symbol $c_\perp$),
(2) the binary set theoretic union operator (function from $\mathcal{D}^2$ to $\mathcal{D}$) to the function symbol *union*,
(3) the characterization of the subsets of $\mathcal{A}$ to the predicate symbol *on*: $on(\mathcal{S})$ if and only if $\mathcal{S} \subseteq \mathcal{A}$,
(4) the binary set theoretic inclusion relation (binary relation on $\mathcal{D}^2$) to the predicate symbol $\subseteq$,
(5) the binary relation of attack between sets of arguments induced by $\mathcal{R}$, and defined by $\mathcal{S}_1 \mathcal{R} \mathcal{S}_2$ if and only if $\mathcal{S}_1 \subseteq \mathcal{A}$, $\mathcal{S}_2 \subseteq \mathcal{A}$ and $\exists x_1 \in \mathcal{S}_1$, $\exists x_2 \in \mathcal{S}_2$, such that $x_1 \mathcal{R} x_2$, to the predicate symbol $\rhd$.

Among the formulae that can be built on the signature $\Sigma_{\mathbf{U}}$, we find the specific axioms of the theory, that will allow to describe AFs on the universe:

Let x, y, z be variables of $\text{YALLA}_{\mathbf{U}}$,

Axioms for set inclusion

- $\forall x\ (c_\perp \subseteq x)$
- $\forall x\ (x \subseteq x)$
- $\forall x, y, z\ ((x \subseteq y \wedge y \subseteq z) \implies x \subseteq z)$

Axioms for set operators

- $\forall x, y\ (x \subseteq union(x, y))$
- $\forall x, y\ (y \subseteq union(x, y))$
- $\forall x, y, z\ (((x \subseteq z) \wedge (y \subseteq z)) \implies (union(x, y) \subseteq z))$

Axioms combining set operators and attack relation

- $\forall x, y, z\ (((x \rhd y) \wedge (x \subseteq z)) \implies (z \rhd y))$
- $\forall x, y, z\ (((x \rhd y) \wedge (y \subseteq z)) \implies (x \rhd z))$
- $\forall x, y, z\ ((union(x, y) \rhd z) \implies ((x \rhd z) \vee (y \rhd z)))$
- $\forall x, y, z\ ((x \rhd union(y, z)) \implies ((x \rhd y) \vee (x \rhd z)))$

Axioms for the predicate *on*

- $on(c_\perp)$
- $\forall x, y\ ((on(x) \wedge (y \subseteq x)) \implies on(y))$
- $\forall x, y\ ((on(x) \wedge on(y)) \implies on(union(x, y)))$
- $\forall x, y\ ((x \rhd y) \implies (on(x) \wedge on(y)))$

An AF belonging to $\Gamma_{\mathbf{U}}$ can be described by its characteristic formula in the language $\text{YALLA}_{\mathbf{U}}$.

Def. 20 (Formula describing an AF). Let the function $\Phi_{\mathbf{U}}$ be defined as follows:

$$\begin{aligned}
\Phi_{\mathbf{U}} : a\Gamma_{\mathbf{U}} &\to \text{YALLA}_{\mathbf{U}} \\
(\mathcal{A}, \mathcal{R}) &\mapsto on(\mathcal{A}) \wedge \\
&\qquad \bigwedge_{x \in \mathcal{A}_{\mathbf{U}} \setminus \mathcal{A}} \neg on(\{x\}) \wedge \\
&\qquad \bigwedge_{(x,y) \in \mathcal{R}} (\{x\} \rhd \{y\}) \wedge \\
&\qquad \bigwedge_{(x,y) \in \mathcal{R}_{\mathbf{U}} \setminus \mathcal{R}} \neg(\{x\} \rhd \{y\})
\end{aligned}$$

$\Phi_{\mathbf{U}}(\mathcal{A}, \mathcal{R})$ is called the *characteristic formula of* $(\mathcal{A}, \mathcal{R})$.[23]

Note that $(\mathcal{A}, \mathcal{R})$ determines the unique structure (Def. 19) which is a model of $\Phi_{\mathbf{U}}(\mathcal{A}, \mathcal{R})$.

Some additional notations are used for encoding the argumentation semantics: let t_1 and t_2 be terms of $\text{YALLA}_{\mathbf{U}}$,

$$t_1 = t_2 \overset{\text{def}}{\equiv} (t_1 \subseteq t_2) \wedge (t_2 \subseteq t_1),$$

$$t_1 \neq t_2 \overset{\text{def}}{\equiv} \neg(t_1 = t_2),$$

$$singl(t_1) \overset{\text{def}}{\equiv} (t_1 \neq c_\perp) \wedge \forall t_2 \left(((t_2 \neq c_\perp) \wedge (t_2 \subseteq t_1)) \implies (t_1 \subseteq t_2) \right).$$

The following property obviously holds: $(\mathcal{A}, \mathcal{R}) \models singl(t)$ if and only if the term t is interpreted by a singleton of $\mathcal{A}$.

Then the principles used in argumentation semantics can be encoded in terms of $\text{YALLA}_{\mathbf{U}}$ formulae:

Prop. 3. *Let $\mathcal{A}_{\mathbf{U}}$ be a set of arguments and $(\mathcal{A}, \mathcal{R})$ be an AF such that $\mathcal{A} \subseteq \mathcal{A}_{\mathbf{U}}$ and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$. Let t, t_1, t_2, t_3 be terms of* $\text{YALLA}_{\mathbf{U}}$.

- *t is* conflict-free *in $(\mathcal{A}, \mathcal{R})$ if and only if $(\mathcal{A}, \mathcal{R}) \models on(t) \wedge (\neg(t \rhd t))$. The latter formula is denoted by $F(t)$.*
- *t_1* defends each element *of t_2 in $(\mathcal{A}, \mathcal{R})$ if and only if $(\mathcal{A}, \mathcal{R}) \models (\forall t_3\ ((singl(t_3) \wedge (t_3 \rhd t_2)) \implies (t_1 \rhd t_3)))$. The latter formula is denoted by $t_1 \rhd\!\rhd t_2$.*
- *t is* admissible *in $(\mathcal{A}, \mathcal{R})$ if and only if $(\mathcal{A}, \mathcal{R}) \models (F(t) \wedge (t \rhd\!\rhd t))$. The latter formula is denoted by $A(t)$.*
- *t is a* complete extension *of $(\mathcal{A}, \mathcal{R})$ if and only if $(\mathcal{A}, \mathcal{R}) \models (A(t) \wedge \forall t_2\ ((singl(t2) \wedge (t \rhd\!\rhd t_2)) \implies (t_2 \subseteq t)))$. The latter formula is denoted by $C(t)$.*
- *t is the* grounded extension *of $(\mathcal{A}, \mathcal{R})$ if and only if $(\mathcal{A}, \mathcal{R}) \models (C(t) \wedge \forall t_2\ (C(t_2) \implies (t \subseteq t_2)))$. The latter formula is denoted by $G(t)$.*

[23] When t denotes a term of $\text{YALLA}_{\mathbf{U}}$, t is identified with the subset of $\mathcal{A}_{\mathbf{U}}$ which interprets t. It is the case for $\{x\}$ for instance.

- *t is a* stable extension *of* $(\mathcal{A}, \mathcal{R})$ *if and only if* $(\mathcal{A}, \mathcal{R}) \models (F(t) \wedge \forall t_2\ ((singl(t_2) \wedge \neg(t_2 \subseteq t)) \implies (t \rhd t_2)))$. *The latter formula is denoted by* $S(t)$.
- *t is a* preferred extension *of* $(\mathcal{A}, \mathcal{R})$ *if and only if* $(\mathcal{A}, \mathcal{R}) \models (A(t) \wedge \forall t_2\ (((t_2 \neq t) \wedge (t \subseteq t_2)) \implies \neg A(t_2)))$. *The latter formula is denoted by* $P(t)$.

Due to the finite size of a universe, each $\text{YALLA}_\mathbf{U}$ formula can be viewed as a propositional formula and the satisfiability problem of a $\text{YALLA}_\mathbf{U}$ base is a NP-complete problem.

In [55] $\text{YALLA}_\mathbf{U}$ is used for expressing update in argumentation dynamics. For instance, in the case of a debate, classical update amounts to consider a formula φ in $\text{YALLA}_\mathbf{U}$ representing a current state of knowledge about exchanged arguments (i.e., it may encompass several possible AFs), and a new piece of information α stating that the debate has evolved in such a way that α now holds (i.e., the current state of the debate is inside a set of AFs satisfying α). Updating φ by α gives a formula $\varphi \diamond \alpha$ that represents the set of AFs corresponding to an evolution of the debate where a change has been done imposing α. Nevertheless, an AF can only evolve by an allowed operation made by an agent (according to the agent's own AF and her target). That means that some transitions are not allowed. So the update operators in argumentation dynamics must take into account these constraints: let $\mathcal{T}$ be a set of authorized transitions, an update operator is a mapping from $\text{YALLA}_\mathbf{U} \times \text{YALLA}_\mathbf{U}$ to $\text{YALLA}_\mathbf{U}$ which associates with any formula φ and any formula α a formula, denoted by $\varphi \diamond_\mathcal{T} \alpha$ satisfying $\mathcal{T}$. In [55], a general update operator is defined following this idea. Then refinements are proposed in order to give a logical translation of previous characterizations proposed in [15,27].

6.1.2. Some examples

Ex. 3.2.4 (cont'd) Let us consider the AF given in Ex. 3.2.4 as the universe $(\mathcal{A}_\mathbf{U} = \{a, b, c\}, \mathcal{R}_\mathbf{U} = \{(a, b), (c, b)\})$. Let $(\mathcal{A}_1, \mathcal{R}_1)$ and $(\mathcal{A}_2, \mathcal{R}_2)$ be the two AFs built the universe $(\mathcal{A}_\mathbf{U}, \mathcal{R}_\mathbf{U})$ defined by: $\mathcal{A}_1 = \{a, b, c\}$, $\mathcal{R}_1 = \{(a, b)\}$, $\mathcal{A}_2 = \{a, b\}$, $\mathcal{R}_2 = \{(a, b)\}$.

Let φ_1 be the formula $on(\{a, b, c\}) \wedge (\{a\} \rhd \{b\})$ and φ_2 be the formula $on(\{a, b\}) \wedge (\{a\} \rhd \{b\})$.

We have $(\mathcal{A}_1, \mathcal{R}_1) \models \varphi_1$, $(\mathcal{A}_1, \mathcal{R}_1) \models \varphi_2$, $(\mathcal{A}_2, \mathcal{R}_2) \models \varphi_2$. However $(\mathcal{A}_2, \mathcal{R}_2)$ is not a model of φ_1, as $\{a, b, c\}$ is not a subset of $\mathcal{A}_2$.

Moreover, following Definition 20, we have:[24]

$$\Phi_\mathbf{U}(\mathcal{A}_1, \mathcal{R}_1) = on\big(\{a, b, c\}\big) \wedge \big(\{a\} \rhd \{b\}\big) \wedge \neg\big(\{c\} \rhd \{b\}\big)$$

$$\Phi_\mathbf{U}(\mathcal{A}_2, \mathcal{R}_2) = on\big(\{a, b\}\big) \wedge \neg\big(on\big(\{c\}\big)\big) \wedge \big(\{a\} \rhd \{b\}\big) \wedge \neg\big(\{c\} \rhd \{b\}\big)$$

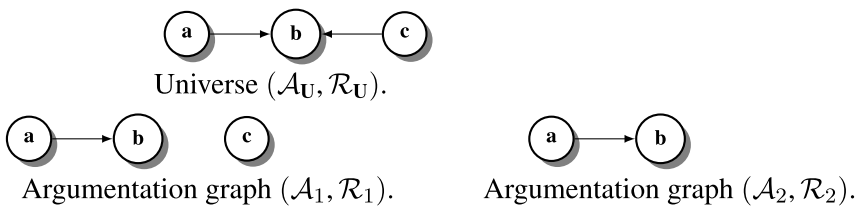


Fig. 1. An example of argumentation graphs built on a universe.

[24]Note that the absence of an attack is expressed only if this attack is in the universe: $\neg(\{c\} \rhd \{b\})$ is in $\Phi_\mathbf{U}(\mathcal{A}_1, \mathcal{R}_1)$ as c attacks b in $\mathbf{U}$, whereas $\neg(\{b\} \rhd \{a\})$ is not in $\Phi_\mathbf{U}(\mathcal{A}_1, \mathcal{R}_1)$ as b does not attack a in $\mathbf{U}$.

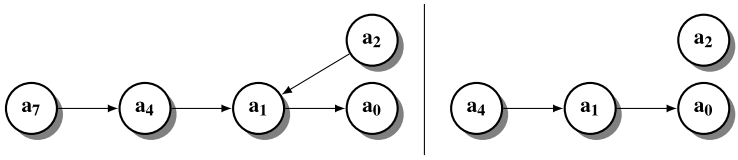


Fig. 2. Two possible cases for the argumentation graph of Agent Ag_b.

More complex assertions can be expressed in YALLA$_{\mathbf{U}}$, such as, for instance, the fact that b does not belong to the grounded extension of $(\mathcal{A}_1, \mathcal{R}_1)$:

$$\Phi_{\mathbf{U}}(\mathcal{A}_1, \mathcal{R}_1) \wedge \nexists t\big(G(t) \wedge \big(\{b\} \subseteq t\big)\big)$$

YALLA$_{\mathbf{U}}$ also enables the expression of incomplete knowledge held by an agent about AFs built on the universe, as shown by the following example:

Ex. 6.1.1 (cont'd) Let us consider the universe $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$ given in the trial example. Assume that Agent Ag_a has only a partial knowledge about the AF built by Agent Ag_b. Indeed, Ag_a hesitates between two possible situations for Ag_b's AF, namely $(\mathcal{A}_1, \mathcal{R}_1)$ and $(\mathcal{A}_2, \mathcal{R}_2)$ given in Fig. 2. So Agent Ag_a has a doubt about the existence of a_7 and of the attacks from a_7 to a_4 and from a_2 to a_1.

The knowledge held by Ag_a can be expressed by the following YALLA$_{\mathbf{U}}$ formula:

$$\begin{aligned}
\varphi = {} & on\big(\{a_0, a_1, a_2, a_4\}\big) \wedge \\
& \neg\big(on\big(\{a_3\}\big)\big) \wedge \neg\big(on\big(\{a_5\}\big)\big) \wedge \neg\big(on\big(\{a_6\}\big)\big) \wedge \\
& \big(\{a_4\} \rhd \{a_1\}\big) \wedge \big(\{a_1\} \rhd \{a_0\}\big) \wedge \\
& \neg\big(\{a_0\} \rhd \{a_1\}\big) \wedge \neg\big(\{a_3\} \rhd \{a_2\}\big) \wedge \neg\big(\{a_5\} \rhd \{a_4\}\big) \wedge \\
& \neg\big(\{a_6\} \rhd \{a_1\}\big) \wedge \\
& \begin{pmatrix} (on(\{a_7\}) \wedge (\{a_7\} \rhd \{a_4\}) \wedge (\{a_2\} \rhd \{a_1\})) \\ \vee \\ (\neg(on(\{a_7\})) \wedge \neg(\{a_7\} \rhd \{a_4\}) \wedge \neg(\{a_2\} \rhd \{a_1\})) \end{pmatrix}
\end{aligned}$$

φ is satisfied by only two structures which correspond to $(\mathcal{A}_1, \mathcal{R}_1)$ and $(\mathcal{A}_2, \mathcal{R}_2)$.
Note that $\varphi \equiv \Phi_{\mathbf{U}}(\mathcal{A}_1, \mathcal{R}_1) \vee \Phi_{\mathbf{U}}(\mathcal{A}_2, \mathcal{R}_2)$.

6.1.3. Related works

Other works propose dedicated logical languages for dealing with dynamics in argumentation frameworks. See for instance [18,44,45,48,51,52]. Moreover, it is worth noticed that a survey has been done in the context of dynamics in argumentation frameworks, see [53].

6.2. The approach by Villata et al.

In [93] is proposed a propositional logic of argumentation to specify and verify requirements in AFs. Note that a modal variant of this logic has been proposed (see Section 10).

The input consists of an AF together with requirements to be satisfied. The outputs are formulae encoding the framework and formulae encoding the requirements. Examples of such requirements are:

"Argument a attacks argument b"; "argument a defends argument b"; "argument a or argument b is acceptable"; "if argument a is accepted (in an extension) then argument b is accepted too (in the same extension) and argument c is not accepted".

The basic ideas of the logic for specification and verification are close to the ideas of YALLA. A model of the logic represents an AF, and such a model satisfies formulae representing the fact that arguments attack or defend each other, or whether sets of arguments are extensions. Moreover the models are built on a given universe of arguments.

Atoms of the language represent sets of arguments. The attack relation is explicitly encoded by a new logical connective $\triangleright$. The formula $p \triangleright q$ is interpreted in $(\mathcal{A}, R)$ as "there is an argument in p that attacks an argument in q", where p, q denote subsets of $\mathcal{A}$. There is also a logical connective for defence, which can be defined in terms of the attack connective, as follows:

$$a \oslash b \equiv \bigwedge_{c \subseteq \mathcal{A}} \big((c \triangleright b) \rightarrow (a \triangleright c)\big)$$

The above connectives allow the specification of requirements related to the structure of an AF.

In order to specify requirements related to the semantics, new kinds of formulae are considered such as $F(p)$, $A(p)$, or $G(p)$ for instance. The formula $F(p)$ (resp. $A(p)$, $G(p)$) is interpreted in $(\mathcal{A}, R)$ as "the set of arguments of $\mathcal{A}$ denoted by p is conflict-free (resp. admissible, the grounded extension) in $(\mathcal{A}, R)$". The semantics are thus considered as primitives of the language.

Note that all the verifications need a model-theoretic approach, as the logic has not been axiomatized. So the verifications have to be made at the AF level.

7. Encoding of extension-based semantics

In this section are reviewed two approaches that, given an extension-based semantics σ, produce a formula Φ_σ whose satisfiability answers the σ-extension problem for the input (usually, an AF and a candidate subset of the arguments). The role of the logic (it is propositional logic) is to specify the semantics but there is no encoding of the graph itself as it is done by the approaches presented in Section 4. Both approaches are theoretical as well as practical: implementations exist.

7.1. The approach by Besnard and Doutre

The original proposal is [11], extended in [12]. Independently, an equivalent proposal is [94].

The idea is as follows. Let σ be an extension-based semantics, $(\mathcal{A}, \mathcal{R})$ be an AF and S be a subset of $\mathcal{A}$. A formula $\Phi(\sigma, S)$ is produced which is satisfiable iff S is a σ-extension of the AF ("Satisfiability approach"). A semantics being defined by a set of principles, a formula associated with each principle is produced, then composing these formulae results in the formula $\Phi(\sigma, S)$.

Logic is used for specifying the principles of the semantics and the produced formula is parametrized by S (and by σ of course).

7.1.1. Description of the approach

For a given $(\mathcal{A}, \mathcal{R})$, a set of propositional symbols $a, b, c, \ldots$ is introduced to represent the elements of $\mathcal{A}$. For simplicity, the same symbol is used, i.e., a is regarded as a propositional symbol whenever $a \in \mathcal{A}$. An argumentative semantics σ can then be mapped to a propositional formula (dependent on a subset S of $\mathcal{A}$) that happens to be satisfiable iff S is a σ-extension of $(\mathcal{A}, \mathcal{R})$.

Such a formula is constructed in view of the interpretation for a propositional symbol a: a is *true* means that a is in the extension.

Also, the usual conditions underlying admissibility in extension-based semantics are captured:

Conflict-free

$$\bigwedge_{a\mathcal{R}b}(a \to \neg b)$$

Admissible

$$\bigwedge_{b\mathcal{R}a}\bigvee_{c\mathcal{R}b} c$$

Complete

$$\left(\bigwedge_{b\mathcal{R}a}\bigvee_{c\mathcal{R}b} c\right) \to a$$

and so on . . .

General test:. $S \subseteq \mathcal{A}$ is a σ-extension of $(\mathcal{A}, \mathcal{R})$ iff the formula below is satisfiable:

$$\bigwedge_{a\in S} a \wedge \bigwedge_{a\in\mathcal{A}\setminus S} \neg a \wedge \Phi_\sigma$$

In the above formula, Φ_σ captures the conditions expressing that S is a σ-extension of $(\mathcal{A}, \mathcal{R})$.

For example, in the formula enabling to determine whether S is a stable extension of $(\mathcal{A}, \mathcal{R})$ there is:

$$\Phi_s = \bigwedge_{a\in\mathcal{A}}\left(a \leftrightarrow \bigwedge_{b\mathcal{R}a} \neg b\right).$$

As to the case of complete extensions,

$$\Phi_c = \bigwedge_{a\in\mathcal{A}}\left[\left(a \to \bigwedge_{a\mathcal{R}b} \neg b\right) \wedge \left(a \leftrightarrow \bigwedge_{b\mathcal{R}a}\bigvee_{c\mathcal{R}b} c\right)\right].$$

7.1.2. Some implementations

Besnard and Doutre's approach has been used in many different implementations. Here we focus on four examples.

In [12,13] and thanks to the project SESAME (see [10]), an implementation has been proposed under the form of a software allowing the definition of semantics with logical formulae (aggregation of formulae, each of them representing a requirement that must be respected by the set of arguments candidate, such as conflict-freeness, or defence).

In [95], the encoding proposed by Besnard and Doutre in [11] is extended in order to take into account the semi-stable semantics. Then two extensions of SAT are used, the minimal correction sets (MCS) and the backbone (BB): consider a CNF propositional formula Φ, the MCS problem consists in the

computation of the minimal sets of clauses issued from Φ such that, after their removal, Φ becomes satisfiable; the BB problem consists in identifying the litterals that are *true* in each model of Φ. The idea of this paper is to establish links between the MCS and BB problems and the computation of semi-stable, eager and ideal extensions.

In [72], the encoding proposed by Besnard and Doutre in [11] is used in two different ways. First, it is used directly in a SAT solver for computing the semantics that are in the first level of the polynomial hierarchy. Secondly, it is extended with some weights for computing the semantics that are in the second or more level of the polynomial hierarchy (with a flavour of constraints programming techniques) then used in a Partial Max-SAT solver (the Partial Max-SAT problem is an optimisation problem that consists in satisfying the more possible clauses of a given formula with respect to the weights given to each clause). These algorithms have been implemented in a software called CoQuiAAS.

In [74], the encoding proposed by Besnard and Doutre in [11] is extended for the computation of three new problems: find an extension that is maximal in terms of cardinality, repair a set of arguments in order to transform it into an extension, and adjust an extension such that it contains (or not) a given argument. Each of these new problems is encoded in logic and solved either by an iteration of SAT solver calls, or by a Max-SAT solver call.

7.2. The CEGARTIX approach

In [59], a specific encoding of some extension-based semantics is proposed, using two propositional symbols for each argument (for an argument a, the symbols are x_a and y_a). For instance, the formula corresponding to the complete semantics is the following one:

For a given $(\mathcal{A}, \mathcal{R})$,

$$\Phi_c = \bigwedge_{(a,b)\in\mathcal{R}} (\neg x_a \vee \neg x_b) \wedge \bigwedge_{(b,a)\in\mathcal{R}} \left(x_a \rightarrow \bigvee_{(c,b)\in\mathcal{R}} x_c \right) \wedge$$
$$\bigwedge_{b\in\mathcal{A}} \left(y_b \leftrightarrow \left(x_b \vee \bigvee_{(c,b)\in\mathcal{R}} x_c \right) \right) \wedge \bigwedge_{a\in\mathcal{A}} \left(\left(\bigwedge_{(b,a)\in\mathcal{R}} y_b \right) \rightarrow y_a \right)$$

As it is said in [59]: "The first line declares the conditions for admissible sets following the definition: any admissible set must be (i) conflict free and (ii) each argument in the set must be defended by the set. The second line declares (i) the value of auxiliary atoms y_a (y_a is *true* iff either x_a is *true* or some x_b is *true* where b attacks a in the AF), and that (ii) each argument a defended by the extension is contained in the extension."

Here is a small illustration to give some insight into the intuition behind the items x_a and y_a; let us prove that the end of the second line of Φ_c implies that each argument defended by the conflict-free extension is contained in the extension:

a is defended by the extension is encoded by $(\bigwedge_{(b,a)\in\mathcal{R}} \bigvee_{(c,b)\in\mathcal{R}} x_c) \rightarrow x_a$.
Let us assume that: $\forall b$ s.t. $(b, a) \in \mathcal{R}$, $\bigvee_{(c,b)\in\mathcal{R}} x_c$.
By definition of y_b (first part of the second line of Φ_c), we have that: $\forall b$ s.t. $(b, a) \in \mathcal{R}$ y_b.
From the end of the second line of Φ_c, we can deduce y_a, that is by definition, $x_a \vee \bigvee_{b\mathcal{R}a} x_b$.
Moreover, from the first part of the first line of Φ_c (conflict-free requirement), we have that $x_a \rightarrow \neg(\bigvee_{(b,a)\in\mathcal{R}} x_b)$.
So we obtain x_a.

Models of Φ_c characterize the complete extensions of $(\mathcal{A}, \mathcal{R})$ in the sense that x_a is *true* in a model I ($x_a \in I$) iff the argument a is in the extension characterized by I.

Then this encoding is used in a software called CEGARTIX (see [58,60]) that computes skeptical and credulous acceptance under given semantics. It is based on NP-oracles (basically the MiniSAT solver).

8. Encoding of labelling-based semantics

The purpose of the works reviewed in this section is to encode labelling-based semantics[25] by a set of logical formulae. These formulae express the different constraints associated with a kind of labelling (complete, grounded, stable, preferred). One of the approaches has led to implemented systems for computing semantics.

8.1. The approach by Caminada and Gabbay

In [25], metalevel approaches are proposed for talking about argumentation. A metalevel approach describes an argumentation framework from "above", using another language and logic. The metalevel language can be classical logic or modal logic. The case of modal logic will be presented in Section 10. Here, we focus on the metalevel approach that uses classical logic for encoding argument labellings.

The logical language includes the equality predicate ("="), a binary predicate R and the three unary predicates Q_0, Q_1 and $Q_?$.

Intuitively, given an AF denoted by $(\mathcal{A}, \mathcal{R})$, and a labelling ℓ, an interpretation of the language can be obtained: $\mathcal{A}$ is taken as the domain, the predicate R is interpreted by the attack relation $\mathcal{R}$ of the AF and ℓ is used to get the interpretation of the predicates Q_0, Q_1 and $Q_?$ as follows: $a \in Q_0$ iff ℓ labels a as `out`, $a \in Q_1$ iff ℓ labels a as `in`; $a \in Q_?$ iff ℓ labels a as `und`.

Let us consider the following classical theory denoted by $\Delta(R, Q_0, Q_1, Q_?)$ (or Δ for short):

(1) $\forall x(Q_0(x) \vee Q_1(x) \vee Q_?(x))$
(2) $\neg\exists x(Q_i(x) \wedge Q_j(x))$ for $i \neq j$, $i, j \in \{0, 1, ?\}$
(3) $\forall y(\forall x(xRy \rightarrow Q_0(x)) \rightarrow Q_1(y))$
(4) $\forall y(\exists x(xRy \wedge Q_1(x)) \rightarrow Q_0(y))$
(5) $\forall y(\forall x(xRy \rightarrow (Q_0(x) \vee Q_?(x))) \wedge \exists x(xRy \wedge Q_?(x)) \rightarrow Q_?(y))$

Any model of the above theory Δ with domain D defines an argumentation framework $(\mathcal{A}, \mathcal{R})$ with $\mathcal{A} = D$, $\mathcal{R} = R$ and a complete labelling ℓ defined from the elements satisfying the predicates Q_0, Q_1 and $Q_?$.

Then other labellings can be characterized as particular models of Δ. For instance, the grounded labelling is obtained with a model that minimizes Q_1, whereas the preferred labellings are obtained with models that maximize Q_1. Note that second-order formulae are needed in order to express the concept "Q_1 is minimal" or "Q_1 is maximal". That leads to the circumscription technique.[26] In short, standard circumscription gives the minimal relation(s) that enjoy a given set of properties (e.g., circumscribing $\forall xy\ R(x, y) \vee R(y, x)$ gives all total antisymmetric relations over the domain). Maximization is obtained dually (although this is usually not called circumscription).

When dealing with a specific AF, additional axioms are needed. They use "=" and constant names for denoting the arguments. Let $\Theta(AF)$ denote the set of axioms describing $AF = (\mathcal{A}, \mathcal{R})$:

[25] Various definitions have been recalled in Section 2 and Section 5.

[26] An example of the use of circumscription in order to solve AF reasoning problems is described in [1].

(1) $\forall x\ (\bigvee_{a\in\mathcal{A}} x = a)$
(2) $\bigwedge_{a,b\in\mathcal{A},a\neq b} a \neq b$
(3) $\bigwedge_{(a,b)\in\mathcal{R}} aRb$

Let $\Delta(AF) = \Delta \cup \Theta(AF)$. It can be shown that characterizing some of the labellings is easier, since for a given AF, the set of arguments is finite. For instance, the grounded labelling of AF can be characterized as the set of x such that $\Delta(AF) \vdash Q_1(x)$. Moreover, as the set of arguments is finite, circumscription becomes first-order.

8.2. *The approach by Cerutti et al.*

In [36], three propositional symbols are defined for each argument: I_a, O_a and U_a meaning that the value of a in the labelling is respectively `in`, `out` or `und`. Moreover, for each value of the labelling, two formulae are given for expressing the necessary and the sufficient conditions corresponding to the assignment of this value. For instance, considering an AF denoted by $(\mathcal{A}, \mathcal{R})$, we have for the value `in`:

$$\left(\bigvee_{b|(b,a)\in\mathcal{R}} \neg O_b\right) \vee I_a \text{ (sufficient condition for } I_a)$$

$$\bigwedge_{b|(b,a)\in\mathcal{R}} (O_b \vee \neg I_a) \text{ (necessary condition for } I_a)$$

Other formulae are defined for expressing some constraints about labellings. For instance, the fact that "an argument has one and only one value in a labelling" is expressed by the formula:

$$\bigwedge_{a\in\mathcal{A}} \big((I_a \vee O_a \vee U_a) \wedge (\neg I_a \vee \neg O_a) \wedge (\neg I_a \vee \neg U_a) \wedge (\neg U_a \vee \neg O_a)\big)$$

or the fact that "unattacked arguments must be `in`" is encoded by:

$$\bigwedge_{a|\nexists b,(b,a)\in\mathcal{R}} I_a$$

Several combinations of these formulae are proposed in order to encode complete labellings.

Note that in the case of a finite AF, the encodings of complete labellings proposed in [36] and [25] can be matched, owing to the equivalence between the alternative definitions of complete labellings provided in [25] (see Section 2.2).

8.3. *Some implementations*

At least three systems have been developed using the approach proposed in [36].

In [36], an algorithm using a SAT solver is proposed for computing the preferred semantics. This algorithm is called PrefSAT.

In [37], a system called ArgSemSAT is proposed including PrefSAT for computing the preferred semantics. It also includes an approach proposed by the same authors using a decomposition of the argumentation graph in its strong connected components (SCC) and the computation of the preferred extensions by a propagation process across these SCCs with a call to a SAT solver for each SCC.

In [9], a software called LabSAT allows the computation of extensions for several semantics (complete, stable, preferred, grounded) and also the resolution of the credulous and skeptical acceptance problems. It uses the encoding proposed in [36].

Moreover, due to the ICCMA competition [16], many other solvers have been developed. See for instance [38–40,62,77] (once again this list is not exhaustive).

9. Argumentation frameworks and logic programming

The close connection between AF semantics and Logic Programming (LP) semantics goes back to Dung's work [54]. Dung introduced a transformation from LP to AF, and showed that stable models (resp. the well-founded model) of a logic program correspond to stable extensions (resp. the grounded extension) of the associated AF. These results have been extended by connections between LP 3-valued stable models (resp. regular models) and complete (resp. preferred) extensions [24,96]. Roughly speaking, the purpose of these works is to provide an argumentative semantics for logic programs. So following our choice presented in Section 1, these works are out of the scope of this survey.

In the same paper [54], Dung also introduced a converse transformation from AF to LP, and showed that stable extensions (resp. the grounded extension) of an AF can be obtained as stable models (resp. the well-founded model) of the associated logic program. These results have been extended to relate other AF semantics to LP semantics [24,26,61,83,84,89]. In that case, the purpose is rather to apply computational techniques of Logic Programming to argumentation and so matches with the topic we are interested in. In this context, note that the relationship between argumentation and logic programming has been the subject of intensive research. Currently, there are different mappings which allow to transform an AF into a logic program, all of them offering different characterizations of argumentation semantics in terms of LP semantics. A summary of these characterizations can be found in [83] with all the associated references. Moreover, some works have also considered the issue of using ASP-solvers to compute the extensions of AFs under different semantics [47,61,69,90].

In the following, we first consider some works relating a (simple) AF and Logic Programs. Then we consider generalizations to enriched argumentation frameworks (BAF, RAF). The works reviewed in this section can be distinguished according to the following features:

- AF semantics can be encoded under the form of extension-based semantics as in [54,84], or under the form of labelling-based semantics (as for instance in [24,89]).
- Different semantics of AF correspond to different semantics of a same logic program as in [24,54] or different semantics of AF are all characterized by the single 2-valued stable model semantics of different transformed programs as in [89].

Note that all the logic programs that we consider below are normal logic programs, that is logic programs whose rules may contain weak negation (i.e. negation as failure, with the symbol *not*) but not strong negation (i.e. the classical negation of classical logic), and where the head of each rule is a single atom. From now, normal logic programs will be called logic programs. Atoms of the form *not a* will be called weak atoms and the weak part of a rule consists in the set of its weak atoms.

9.1. From an AF to a logic program

The input is an AF. The output consists of a logic program P together with the characterization of some of the standard argumentation semantics (at least the grounded semantics and the stable semantics) using the models of P (two-valued or three-valued models depending on the considered approach).

The logical formalism enables to encode attacks. Moreover, depending of the approaches, the concept of attack is encoded in an explicit way (as for instance in [26,54,84]) or in an implicit way (as for instance in [24]).

Two kinds of work can be distinguished. The first kind lies in the spirit of Dung's work and concerns works using two-valued models for LP semantics. The second kind of works uses three-valued models for LP semantics.

Towards logic programs under 2-valued semantics. Let us start by the initial approach of [54]. Given an AF $(\mathcal{A}, \mathcal{R})$, the associated logic program consists of three parts:

- The rules describing the attacks in $\mathcal{R}$: $\{attack(x, y) \leftarrow |(x, y) \in \mathcal{R}\}$.
- The rules defining acceptability: $acc(x) \leftarrow not\ d(x)$.
- The rules defining defeat: $d(x) \leftarrow attack(y, x), acc(y)$

where $acc(x)$ stands for "the argument x is acceptable" and $d(x)$ stands for "the argument x is defeated".

The approaches described in [26,83,84] propose an alternative transformation considering not only attackers but also defenders. The common idea is to encode attack in an explicit way through a literal $def(x)$ meaning "x cannot belong to an admissible set". So the logic program P contains rules that define $def(x)$ given the attackers of x.

The characteristic features of these approaches are:

- The logic program includes weak negation (i.e. negation as failure, with the symbol *not*).
- The encoding captures both conflict-freeness and admissibility principles.
- The predicate *def* is used. $def(x)$ means "x is defeated" or equivalently "x cannot belong to an admissible set".
- The logic program P contains two parts P^- and P^+: P^- encodes conflict-freeness and P^+ encodes admissibility.
  - P^- contains rules of the form $def(x) \leftarrow not\ def(y)$ if y attacks x. So, P^- defines $def(x)$ given the attackers of x.
  - P^+ contains rules of the form $def(x) \leftarrow def(z_1), \ldots, def(z_n)$ if $z_1, \ldots, z_n$ are all the attackers of some y attacker of x. Such a rule defines $def(x)$ given the defenders of x against y.
- Then, acceptability can be encoded through the predicate *acc*, where $acc(x)$ means "x can be considered as accepted", with the rule $acc(x) \leftarrow not\ def(x)$.
- There are correspondences between standard argumentation semantics (grounded, stable, preferred, complete) and two-valued models of the logic program (well-founded, stable, p-stable, supported models).

Note that acceptability is defined by default. That is close to the reinstatement principle. In particular, if an argument x is not attacked, it does not appear as the head of any rule, so x will be accepted.

Ex. 3.2.3 (cont'd) We consider here the AF corresponding to a sequence of two attacks (a attacks b that in turn attacks c). The associated logic program is:

$$def(b) \leftarrow not\ def(a)$$

$$def(c) \leftarrow not\ def(b)$$

$$def(b) \leftarrow \top$$

$def(c) \leftarrow def(a)$

$acc(x) \leftarrow not\ def(x)$ (x being a variable)

Ex. 3.2.4 (cont'd) We consider here the AF corresponding to an argument attacked by two other arguments (a and c attack b). The associated logic program is:

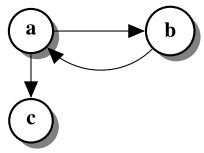
$def(b) \leftarrow not\ def(a)$

$def(b) \leftarrow not\ def(c)$

$def(b) \leftarrow \top$

$acc(x) \leftarrow not\ def(x)$ (x being a variable)

Ex. 9.1.1. Consider the following AF:



The associated logic program is:

$def(b) \leftarrow not\ def(a)$

$def(a) \leftarrow not\ def(b)$

$def(c) \leftarrow not\ def(a)$

$def(b) \leftarrow def(b)$

$def(a) \leftarrow def(a)$

$def(c) \leftarrow def(b)$

$acc(x) \leftarrow not\ def(x)$ (x being a variable)

Ex. 3.4.1 (cont'd) The associated logic program is:

$def(a) \leftarrow not\ def(b)$

$def(b) \leftarrow not\ def(a)$

$def(c) \leftarrow not\ def(b)$

$def(c) \leftarrow not\ def(e)$

$def(d) \leftarrow not\ def(c)$

$def(e) \leftarrow not\ def(d)$

$def(a) \leftarrow def(a)$

$def(b) \leftarrow def(b)$

$def(c) \leftarrow def(a)$

$def(c) \leftarrow def(d)$

$def(d) \leftarrow def(b), def(e)$

$def(e) \leftarrow def(c)$

$acc(x) \leftarrow not\ def(x)$ (x being a variable)

The unique stable model is $\{def(a), def(c), def(e), acc(b), acc(d)\}$. That characterizes the unique stable extension of the AF $\{b, d\}$.

The approach of [89] is rather different in the sense that it proposes several transformations of an AF in different logic programs, each one encoding a labelling-based semantics. These semantics are all characterized by the single 2-valued stable model semantics.

Given an AF, each logic program consists of two parts:

- the rules encoding admissibility, which will belong to all the different programs,
- the rules specific to each semantics.

For instance, let us give the program representing the grounded semantics. The first part contains rules defined as follows:[27]

- $in(x) \leftarrow out(y_1), \ldots, out(y_n)$, where $y_1, \ldots, y_n$ are all the attackers of x.
- $out(x) \leftarrow in(y)$ for each y attacker of x.
- $\leftarrow in(x), not\ out(y)$ for each y attacker of x.
- $\leftarrow out(x), not\ in(y_1), \ldots, not\ in(y_n)$, where $y_1, \ldots, y_n$ are all the attackers of x.

The part specific to the grounded semantics contains rules of the form: $und(x) \leftarrow not\ in(x), not\ out(x)$.

Ex. 3.2.3 (cont'd) The associated logic program under the grounded semantics is:

$in(c) \leftarrow out(b);$	$in(b) \leftarrow out(a);$	$in(a) \leftarrow;$
$out(b) \leftarrow in(a);$	$out(c) \leftarrow in(b);$	
$\leftarrow in(c), not\ out(b);$	$\leftarrow in(b), not\ out(a);$	
$\leftarrow out(c), not\ in(b);$	$\leftarrow out(b), not\ in(a);$	$\leftarrow out(a);$

$und(x) \leftarrow not\ in(x), not\ out(x)$ (for $x \in \{a, b, c\}$).

The unique stable model is $\{in(a), out(b), in(c)\}$.

[27] Note that the rules with an empty head represent constraints.

Ex. 3.2.4 (cont'd) The associated logic program under the grounded semantics is:

$in(c) \leftarrow;$	$in(b) \leftarrow out(a), out(c);$	$in(a) \leftarrow;$
$out(b) \leftarrow in(a);$	$out(b) \leftarrow in(c);$	
$\leftarrow in(b), not\ out(a);$	$\leftarrow in(b), not\ out(c);$	
$\leftarrow out(c), not\ in(b);$	$\leftarrow out(b), not\ in(a), not\ in(c);$	
$\leftarrow out(a);$	$\leftarrow out(c);$	

$und(x) \leftarrow not\ in(x), not\ out(x)$ (for $x \in \{a, b, c\}$).

The unique stable model is $\{in(a), out(b), in(c)\}$.

Ex. 9.1.1 (cont'd) The associated logic program under the grounded semantics is:

$in(a) \leftarrow out(b);$	$in(b) \leftarrow out(a);$	$in(c) \leftarrow out(a);$
$out(a) \leftarrow in(b);$	$out(b) \leftarrow in(a);$	$out(c) \leftarrow in(a);$
$\leftarrow in(a), not\ out(b);$	$\leftarrow in(b), not\ out(a);$	$\leftarrow in(c), not\ out(a);$
$\leftarrow out(a), not\ in(b);$	$\leftarrow out(b), not\ in(a);$	$\leftarrow out(c), not\ in(a);$

$und(x) \leftarrow not\ in(x), not\ out(x)$ (for $x \in \{a, b, c\}$).

The unique stable model is $\{und(a), und(b), und(c)\}$.

Towards logic programs under 3-valued semantics. Let us consider the approach described in [24]. The idea is to encode attack in an implicit way, that is as a rule of the logic program, without any additional predicate. Each argument generates a rule with the name of the argument as its head and the name of the attackers in the weak part of its body. The characteristic features of this approach are:

- No predicate is needed.
- Each argument becomes an atom of the program P.
- The logic program includes weak negation (i.e. negation as failure, with the symbol *not*).
- The logic program contains rules of the form $x \leftarrow not\ y_1, \ldots, not\ y_n$, where $y_1, \ldots, y_n$ are all the attackers of x.
- The logic program is a simple program, that is a program with at most one rule with head x for each argument x. Moreover for each rule of the program the body only contains weak atoms (atoms of the form *not a*).
- There are correspondences between standard argumentation semantics (complete, stable, grounded, preferred) and three-valued models (3-valued stable, 2-valued stable, well-founded, regular) of the logic program.

Ex. 3.2.3 (cont'd) The associated logic program is:

$b \leftarrow not\ a$

$c \leftarrow not\ b$

$a \leftarrow$

Ex. 3.2.4 (cont'd) The associated logic program is:

$b \leftarrow not\ a, not\ c$

$a \leftarrow$

$c \leftarrow$

Ex. 9.1.1 (cont'd) The associated logic program is:

$b \leftarrow not\ a$

$a \leftarrow not\ b$

$c \leftarrow not\ a$

This logic program has three 3-valued stable models: $(\varnothing, \varnothing)$, $(\{a\}, \{b, c\})$, $(\{b, c\}, \{a\})$ corresponding respectively to the three complete extensions of the AF: $\varnothing$, $\{a\}$ and $\{b, c\}$.

Ex. 3.4.1 (cont'd) The associated logic program is:

$b \leftarrow not\ a$

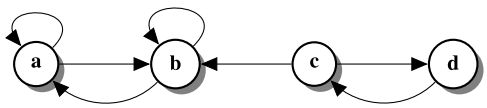
$a \leftarrow not\ b$

$c \leftarrow not\ b, not\ e$

$d \leftarrow not\ c$

$e \leftarrow not\ d$

Ex. 9.1.2. Consider the following AF:



The associated logic program is:

$a \leftarrow not\ a, not\ b$

$b \leftarrow not\ b, not\ a, not\ c$

$c \leftarrow not\ d$

$d \leftarrow not\ c$

9.2. Generalizations to enriched frameworks

Now we consider generalizations to enriched argumentation frameworks: a framework accounting for both attack and support interactions, called AFN, and a framework accounting for higher-order interactions.

9.2.1. AFNs and logic programs

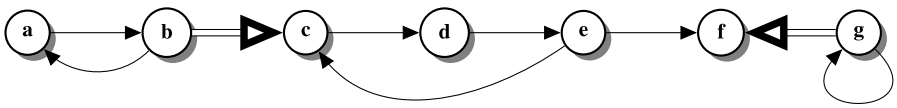
Let us briefly review the work done by F. Nouioua (unpublished report, private communication). That work examines the connections between an Argumentation Framework with Necessities (AFN) and logic programs under three-valued semantics. Both directions are considered but following our choice we discuss only the case when an AFN is encoded into a logic program.

An AFN [78] is a kind of BAF, where the support is collective and is interpreted as a "necessary support": Given E a non-empty set of arguments and a an argument, E is a "necessary support" for a means that the acceptance of a requires the acceptance of *at least one argument* of E. Note that in AFN semantics, acyclicity of the support relation is required among accepted arguments. In other words, in a given extension, support for each argument is provided by at least one of its necessary arguments and there is no risk of deadlock due to necessity cycles.

An AFN is encoded into a logic program as follows:

- For each argument a, an atom a and a rule r_a of head a are created.
- For each set of arguments E that necessary supports an argument, an atom e is created.
- The body of the rule r_a contains all the weak atoms *not* b_i, with b_i attacks a, and all the strong atoms e_j, with E_j supports a.
- For each support E, there is a rule $e \leftarrow x$ for each $x \in E$.
- There are correspondences between AFN labelling-based semantics and three-valued semantics of the program which is obtained.

Ex. 9.2.1. Consider the following AFN:



The associated logic program is:

$a \leftarrow not\ b$

$b \leftarrow not\ a$

$c \leftarrow e_1, not\ e$

$d \leftarrow not\ c$

$e \leftarrow not\ d$

$f \leftarrow e_2, not\ e$

$g \leftarrow not\ g$

$e_1 \leftarrow b$

$e_2 \leftarrow g$

This program has three 3-valued stable models: $(\varnothing, \varnothing)$, $(\{b, e_1\}, \{a\})$, $(\{a, d\}, \{b, c, e, e_1\})$ corresponding respectively to the three complete labellings of the AFN.

9.2.2. Argumentation frameworks with higher level attacks and logic programs by Gabbay

In [65] higher level extended argumentation frameworks are considered, where attacks on other attacks are allowed, at any level. A translation from a higher level framework into a logic program is suggested, as follows:

- The atoms of the logic program are all the arguments and all the attacks.
- For each e, argument or attack, if e is not attacked, the rule $e \leftarrow$ is created.
- Assume that e is attacked by the arguments $a_1, \ldots, a_k$ through the attacks named $\alpha_1, \ldots, \alpha_k$. Assume that each α_i is itself attacked by the arguments $b_i^1, \ldots, b_i^{k(i)}$ through the attacks named $\beta_i^1, \ldots, \beta_i^{k(i)}$. Then the following formula is created: $e \leftarrow \bigwedge_{i=1,k}(not\ a_i \vee \bigvee_{j=1,k(i)}(\beta_i^j \wedge b_i^j))$.
- The above formula must be turned into several clauses in order to get a normal logic program.

Gabbay, in [65], proposes to define the semantics of a higher level extended argumentation framework through known semantics for logic programs, using the above transformation.

However, to the best of our knowledge, this way was not pursued by the author. Instead, Gabbay in [66] proposes to give up the logic programming approach for defining the semantics of higher level argumentation frameworks. Indeed, the new approach consists in rewriting higher level attacks as frameworks with a special kind of collective attacks (called joint attacks).

Note that another connection could be considered between another variant of higher level extended argumentation frameworks, called REBAF, and logic programs (private communication by J. Fandinno). A REBAF [29] is a generalization of AF that allows the representation of both recursive attacks and evidential supports. It is also a generalization of EBAF (see Section 2) with attacks and supports targeting other attacks or supports. Nevertheless, this connection is not in the scope of this survey since this is the encoding of a logic program into a REBAF and, up to now, the other direction (from a REBAF to a logic program) has not been considered.

9.3. Some implementations

Some implementations using logic programming have been proposed. The first one uses a translation of an AF into a logic program. The other ones do not propose a translation of an AF into a logic program but use logic programming for computational issues.

ASPARTIX (Answer Set Programming Argumentation Reasoning Tool) supports reasoning in AFs using the ASP formalism (see [57] for the description of the tool and many references).

The core of ASPARTIX handles AFs. It provides encodings for computing extensions or performing credulous/skeptical reasoning in AFs. A broad range of argumentation semantics is dealt with.

ASPARTIX also handles several other frameworks built on top of AFs, including for instance AFs augmented with preferences (PAFs), BAFs, AFRAs, SETAFs (AFs where attacks are carried by sets of arguments).

[46] presents dialectical proofs for the credulous acceptance problem in constrained argumentation frameworks (CAF, see Section 3.4). A CAF is a generalization of AF that allows additional constraint on arguments to be taken into account in the definition of admissible sets of arguments. A dialectical proof is formalized by a dialogue between two players, the proponent and the opponent. Dialectical proofs are computed by an ASP program which consists of facts encoding a CAF and rules encoding what a dialectical proof is (that is the legal-move function of the dialogue).

The models of the logic program correspond to the dialectical proofs of the CAF (see [73] for the description of the ASP solver ASPeRiX and its application to the credulous acceptance problem in a CAF).

[88] proposes a Boolean algebra to encode acceptability semantics for AFs. A subset of arguments is represented by a Boolean vector and the attack relation is represented by a Boolean matrix. Then series of Boolean operations on vectors and matrices are introduced so that acceptability semantics (namely the admissible, stable and complete semantics) can be encoded by Boolean constraints. These constraints are translated into logic programs and solved using a Constraint Logic Programming over Boolean variables (CLPB) system, which is an instance of the general CLP scheme that extends logic programming with reasoning over Boolean domains. The implementation uses SWI-Prolog, a Prolog system equipped with a CLPB system. Experiments have been conducted, with a comparison with the approach using the solver ArgSemSAT [37] (see Section 8.3).

10. Argumentation frameworks and modal logic

As outlined in [25], there exist several methods for expressing argumentation in modal logic, among which the object-level approach and the meta-level approach. Roughly speaking, in the object-level approach, an AF and its logical translation share the same language (each argument becomes a logical atom), whereas a meta-level approach talks about an AF from "above", using another language and logic (for instance a modal logic).

We first recall some modal logic background. Then, we present three examples of meta-level approaches. In the first two, the input consists of an AF together with a labelling. The output consists of modal formulae that express the characteristic properties of complete and stable semantics.

Finally, we briefly present an object-level approach.

10.1. Modal logic preliminaries

The modal logic K is a propositional system with the modal operator $\Box$ (and its dual $\Diamond$), the usual logical connectives, the symbols $\top$, $\bot$ and atomic propositions $q_1, q_2, \ldots$.

Models for K have the form (S, R, h) where $S \neq \varnothing$ is the set of possible worlds, $R \subseteq (S \times S)$, and h is the assignment function giving for each atomic proposition q a subset $h(q)$ of S.

Satisfaction is defined as follows:

- $t \vDash q$ iff $t \in h(q)$
- $t \vDash (A \wedge B)$, $(\neg A)$, $(A \vee B)$, $(A \rightarrow B)$ as usual
- $t \vDash (\Box A)$ iff for all s such that tRs, $s \vDash A$
- A holds in (S, R, h) iff for all $t \in S$, $t \vDash A$

The system K can be axiomatized as follows:

- propositional tautologies
- $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$
- If $\vDash A$ then $\vDash \Box A$

10.2. The approach by Grossi

In [70] a well-known modal logic (the extension of K with universal modality) is used to formalize basic notions of argumentation theory.

Let $AF = (\mathcal{A}, \mathcal{R})$. AF is viewed as a modal frame, where the set of possible worlds is the set of arguments $\mathcal{A}$ and the accessibility relation is the inverse of the attack relation (intuitively the "being attacked" relation).

An assignment I on $(\mathcal{A}, \mathcal{R})$ is a function from a set of propositional atoms to the subsets of arguments.[28] The fact that an argument a belongs to $I(p)$ can be interpreted as "argument a has property p".

An argumentation model has the form (AF, I) where I is an assignment on AF. As an example of assignment, we find the labellings: *A labelling function ℓ on $\mathcal{A}$ can be viewed as an assignment on the propositional symbols* $1, 0, ?$ *(intuitively* `in`, `out`, `und`*).*

The following statements are interesting in argumentation theory:

- "argument a is attacked by the set of arguments E";
- "argument a is defended by the set of arguments E";
- "the set of arguments E attacks an attacker of argument a".

In order to express these statements, two modal operators are used:

- $\langle\leftarrow\rangle$ whose intuitive reading is "there exists an argument attacking the current one such that", and
- $\langle\forall\rangle$ whose intuitive reading is "there exists an argument such that".

As usual, there are also the dual operators $[\leftarrow]$ and $[\forall]$.

The semantics is as follows: In a given model (AF, I):

- $a \vDash \langle\leftarrow\rangle\phi$ iff there exists s such that s attacks a in AF and $s \vDash \phi$.
- $a \vDash \langle\forall\rangle\phi$ iff there exists s such that $s \vDash \phi$.

Consider for instance the particular case of a labelling. Let a be an argument. $a \vDash \langle\leftarrow\rangle 1$ reads "there exists at least one attacker of a labelled `in`".

Given an argumentation model (AF, I), the statement "argument a is attacked by an argument in the set of arguments of $\mathcal{A}$ that satisfy the property ϕ" can be encoded as $a \vDash \langle\leftarrow\rangle\phi$. As $s \vDash \phi$ reads "s belongs to the set of arguments in $\mathcal{A}$ that satisfy ϕ", it follows that $a \vDash \langle\leftarrow\rangle\phi$ encodes the statement "argument a is attacked by the set of arguments in $\mathcal{A}$ that satisfy ϕ". Similarly, the statement $a \vDash \langle\leftarrow\rangle\langle\leftarrow\rangle\phi$ encodes the statement "argument a is defended by the set of arguments in $\mathcal{A}$ that satisfy ϕ".

The modal logic that is obtained is an extension of the modal logic K, denoted by $K^{\forall}$. It enables to capture basic principles of argumentation semantics, as for instance:

- $[\forall](\phi \rightarrow [\leftarrow]\neg\phi)$ encodes that $[\![\phi]\!]$ [29] is conflict-free.
- $[\forall](\phi \rightarrow [\leftarrow]\langle\leftarrow\rangle\psi)$ encodes that $[\![\phi]\!]$ is acceptable w.r.t. $[\![\psi]\!]$.
- $[\forall](\phi \leftrightarrow [\leftarrow]\neg\phi)$ encodes that $[\![\phi]\!]$ is stable.

Consider the particular case of a labelling. A labelling ℓ is a complete labelling for AF iff the model (AF, ℓ) satisfies the following formula:

$$[\forall]((1 \leftrightarrow [\leftarrow]0) \wedge (0 \leftrightarrow \langle\leftarrow\rangle 1) \wedge \mathit{Label})$$

[28] That is, the codomain of I is the powerset of $\mathcal{A}$.

[29] $[\![\phi]\!]$ denotes the set of arguments in AF that satisfy ϕ.

where *Label* is the formula $(1 \wedge \neg 0 \wedge \neg ?) \vee (\neg 1 \wedge 0 \wedge \neg ?) \vee (\neg 1 \wedge \neg 0 \wedge ?)$, meaning that each argument can get at most one label. See an example given in Section 12.2.

Then model-checking can be used in order to determine whether a given formula is conflict-free, admissible, stable.

Furthermore, Grossi, in [70], presents a game-theoretic proof procedure based on model-checking games for the logic $K^{\forall}$. In such a game, a proponent tries to prove that $a \models \phi$ holds in a given model (AF, I) while an opponent tries to disprove it.

Note that an additional modal machinery including a least fixed point operator is needed to capture the notion of grounded extension.

10.3. The meta-level approach of Caminada and Gabbay

The meta-level approach of [25] is close to the approach of [70]. It can be summarized as follows. Given $AF = (\mathcal{A}, \mathcal{R})$,

- Arguments are viewed as possible worlds; so AF becomes a modal frame for the modal logic K.
- The attack relation becomes the accessibility relation.
- A labelling becomes an assignment of three propositional atoms q_0, q_1, $q_?$.

More precisely, given ℓ a labelling on AF, the associated assignment I is defined as follows:

$$t \in I(q_0) \text{ (or } t \models q_0) \text{ iff } \ell(t) = \texttt{out};$$

$$t \in I(q_1) \text{ (or } t \models q_1) \text{ iff } \ell(t) = \texttt{in};$$

$$t \in I(q_?) \text{ (or } t \models q_?) \text{ iff } \ell(t) = \texttt{und}.$$

The modality $\Box$ means "being attacked by", namely $a \models \Box\phi$ iff for all s such that s attacks a in AF, $s \models \phi$. As usual, $\Diamond$ denotes the dual modality.

Note that the modality $\Box$ corresponds to the modality $[\leftarrow]$ of [70]. However there is no modality corresponding to $[\forall]$.

Complete labellings can be characterized with a set of axioms including for instance:

$\Box q_0 \rightarrow q_1$ (if all attackers of t are `out` then t is `in`)

$\Diamond q_1 \rightarrow q_0$ (if t is attacked by an argument which is `in` then t is `out`).

Then, stable and complete extensions are characterized by the following equations. Let E be a propositional letter denoting a set of arguments (see an example given in Section 12.2).

- E is stable iff $E = \Box\neg E$.
- E is a complete extension iff $E = \Box(\neg E \wedge \Diamond E)$.

The extensions can be obtained as fixed point solutions for the above equations.

10.4. The modal setting of Villata et al.

The purpose of the work reported in [93] is to define a logic for specifying and verifying requirements for AFs. A propositional variant has been presented in Section 6. Here we consider the modal variant,

which allows the expression that some semantics admit multiple extensions and also the expression of properties of the attack relation (such as irreflexivity, or symmetry for instance).

The main difference with the modal approaches of [25] and [70] is that these works describe semantics in the modal language, whereas [93] considers semantics as primitives of the language.

The approach of [93] presents the following features:

- Sets of arguments are viewed as possible worlds.
- The attack relation is interpreted as an accessibility relation among worlds. That implies that the attack relation represents a collective attack (a set of arguments taken together may attack another set of arguments).
- There are two modal operators: $\Box_1$ and $\Box_2$. The modality $\Box_1$ means "attacks", namely $p \models \Box_1\phi$ iff for all q such that p attacks q in the AF, $q \models \phi$. $\Box_2$ is a universal modality (as [∀] in [70]).

As in the propositional variant, primitives of the modal language (such as $F(p)$, $A(p)$) represent semantics. Moreover, in order to abbreviate formulae, a connective for collective attack is defined as follows: $p \rhd q \equiv \Box_2(p \to \Box_1\neg q)$.

The modal logic that is obtained enables to express characterizations such as:

- For any pair (p, q) of sets of arguments such that p attacks q, they cannot be subsets of a conflict-free extension: $\Box_2(p \rhd q) \to F(\neg p \vee \neg q)$
- The grounded semantics admits a single extension: $\models (G(p) \wedge G(q)) \to \Box_2(p \leftrightarrow q)$

10.5. The object-level approach of Caminada and Gabbay

The object-level approach of [25] can be summarized as follows. Given $AF = (\mathcal{A}, \mathcal{R})$,

- Arguments are viewed as atomic propositions in the modal provability logic LN_3.
- The content of AF is represented by a formula $M(AF)$ of the logic LN_3.
- The possible world models of $M(AF)$ are in one-to-one correspondence with the labellings of AF.

The modal formula $M(AF)$ is as follows:

$$M(AF) = \begin{pmatrix} (G((\Box\bot) \vee (\bigwedge_{y \text{ has attackers } y_i}(y \leftrightarrow \bigwedge_i \Diamond\neg y_i)))) \\ \bigwedge \\ (\bigwedge_{x \text{ is not attacked}} Gx) \end{pmatrix}$$

where GA stands for $A \wedge \Box A$.

The provability logic LN_3 has the following axioms and rules:

(1) axioms and rules of modal logic K
(2) $\Diamond A \to \Diamond(A \wedge \Box\neg A)$
(3) $\Box A \to \Box\Box A$
(4) $(\Diamond A \wedge \Diamond B) \to \Diamond(A \wedge B) \vee \Diamond(A \wedge \Diamond B) \vee \Diamond(B \wedge \Diamond A)$
(5) $\Box\Box\Box\bot$

11. Translation of an AF into intuitionistic logic

The translation of an AF into an intuitionistic logic theory has been first considered in [68]. The idea is to use intuitionistic negation to model an attack. The intuitionistic models of the obtained theory characterize the complete extensions.

More recently, [63] presents a translation of an AF into Nelson's constructive logic, an extension of intuitionistic logic including the notion of strong negation as a means to deal with constructive falsity. This logic allows to capture an AF under the stable semantics, at the object level, in the sense that arguments in AF become atoms in the corresponding logical theory and interactions between arguments are expressed by logical connectives. Moreover the translation allows to deal with enriched argumentation frameworks such as frameworks with collective attacks, and frameworks with attacks and evidential supports.

11.1. Background about intuitionistic logic

Let us recall that:

- Intuitionistic implication $\phi_1 \rightarrow \phi_2$ can be understood as a means to construct a proof of the truth of ϕ_2 in terms of a proof of the truth of ϕ_1.
- Strong negation $\sim \phi$ can be understood as the existence of a proof of the falsity of ϕ.
- Intuitionistic negation is defined as $\neg\phi \equiv (\phi \rightarrow \bot)$. It can be understood as a means to obtain a proof of a contradiction from a proof of the truth of ϕ (or roughly speaking as "there cannot be a proof of the truth of ϕ").

Then, a new implication connective, $\Rightarrow$, allows the formalization of the "non contradictory" inference principle (NC): "no belief can be held based on contradictory evidence".

$$\phi_1 \Rightarrow \phi_2 \equiv (\neg \sim \phi_1 \wedge \phi_1) \rightarrow \phi_2$$

Intuitively, $\phi_1 \Rightarrow \phi_2$ can be understood as a means to construct a proof of the truth of ϕ_2 given a proof of the truth of ϕ_1 and the fact that there cannot be a proof of its falsity (or in other words given a "consistent" proof of ϕ_1).

11.2. Abstract argumentation and intuitionistic logic

In [63], intuitionistic logic is used for translating an AF or an EBAF.

Translation of an AF. The translation presented in [63] relies on the following intuition: under the constructive logic point of view, an attack is a "means to construct a proof of the falsity of the attacked argument based on the acceptability of the attacker". Moreover, the acceptability of ϕ is identified with having a consistent proof of it (i.e. there is a proof of the truth of ϕ and there cannot be a proof of the falsity of ϕ).

Then the above intuition of the notion of attack is formalized by a new connective:

$$\phi_1 \rightsquigarrow \phi_2 \equiv \phi_1 \Rightarrow \sim \phi_2$$

In other words, $\phi_1 \rightsquigarrow \phi_2$ says that the acceptability of ϕ_1 allows to construct a proof of the falsity of ϕ_2, and a proof of the falsity of ϕ_2 is identified with ϕ_2 being defeated.

Given $AF = (\mathcal{A}, \mathcal{R})$, the associated theory in constructive logic is:

$$C(AF) \equiv \mathcal{A} \cup \left\{a \rightsquigarrow b | (a, b) \in \mathcal{R}\right\}$$

[63] proves that there is a one-to-one correspondence between the stable extensions of AF and the equilibrium models of the theory $C(AF)$, where the equilibrium models of a theory are a particular selection of constructive logic models of this theory.

Regarding how other extension-based semantics could be characterized in constructive logic is an open topic.

Translation of evidential argumentation frameworks. In the case of an argumentation framework with collective attacks and evidential supports (EBAF) the translation uses the connectives $\bigwedge$, $\rightsquigarrow$ and $\Rightarrow$.

Let $EBAF = (\mathcal{A}, \mathcal{R}, \mathcal{E}, \mathcal{P})$. Recall that the attack relation $\mathcal{R}$ is a subset of $2^{\mathcal{A}} \setminus \varnothing \times \mathcal{A}$, the support relation $\mathcal{E}$ is a subset of $2^{\mathcal{A}} \setminus \varnothing \times \mathcal{A}$, and $\mathcal{P} \subseteq \mathcal{A}$ is the set of distinguished prima-facie arguments (see Section 2.1).

Using the notation $\bigwedge B$ that denotes "the conjunction of the elements of B", the associated theory in constructive logic is:

$$C(EBAF) \equiv \mathcal{P} \cup \left\{ \bigwedge B \rightsquigarrow b \middle| (B, b) \in \mathcal{R} \right\} \cup \left\{ \bigwedge B \Rightarrow b \middle| (B, b) \in \mathcal{E} \right\}$$

Note that the support from the set of arguments B to an argument b is translated by the formula $\bigwedge B \Rightarrow b$ which says that the acceptability of B enables to construct a proof of the truth of b.

As for the case of an AF, in [63] it is proved that there is a one-to-one correspondence between the stable extensions of an EBAF (defined in [29]) and the equilibrium models of the theory $C(EBAF)$.

12. Analysis and conclusion

In this section, we first compare the reviewed approaches according to several criteria. Then, we illustrate the associated encodings on an example. And finally we will conclude this survey.

12.1. Comparison criteria

Recall that our initial choice was to identify and to compare approaches that "use logic for doing argumentation".[30]

At least four criteria can be used:

(1) The first comparison criterion is about the *input* taken into account by the different approaches. In each case, the input consists at least of a graph. Except in the ADF approach, this graph usually represents an AF. Moreover, in some approaches, the AF is extended into either a bipolar AF, or a recursive AF, or an AF with collective interactions, or with weighted arguments/interactions, or with preferences, or built on a given universe.
In some approaches, an additional input is needed: either constraints or requirements that must be satisfied by the output or "candidates" that must be studied in order to satisfy some properties. Table 1 synthetizes all these cases.

(2) The second criterion is about the *aim* of the approaches that use logic for doing argumentation and is related to the produced output. Two kinds of aim can be encountered:

[30]The other direction could be the object of another survey that would cover the relationship between logical frameworks and formal argumentation, including structured argumentation.

Table 1

The first comparison criterion: the input

Input graph	Approaches
Case of an input graph that represents an AF	
Dung	[4,5,11,12,24–26,30,36,43,49,50,52,54,55,59,65,67,70,83,84,89,93]
Bipolar	[63]
Recursive	[34,65]
Weighted	none
With preference	[56]
With collective interactions	[55,63]
On a universe	[55]
Case of an input graph that does not represent an AF	
Dependence graph	ADF approach [20,21]

Other input	Approaches
Acceptance conditions	• (on each argument) ADF approach [20,21]: depending on the chosen acceptance conditions, the dependence graph can represent any type of AF, from Dung AFs to AFs with collective interactions
	• (on the AF) Constrained AF [43]: the acceptance condition is used only for removing some extensions
Requirements	On the AF or the semantics [93]: the requirements are used for constraining the structure of the AF or the resulting labellings
Candidates	Are these candidates actual extensions or labellings for the given semantics? [11,12,25,36,59,70]

Table 2

The second comparison criterion: the aim

Aim		Approaches
Translation of an AF		[24,26,30,34,52,55,56,67,84]
Encoding of semantics	Case of extension-based semantics	[11,12,24–26,30,34,52,54–56,59,63,65,67,78,83,84,89,93]
	Case of labelling-based semantics	[4,5,25,25,36,49,50,70]

- Either the aim is to obtain a logical encoding of the input AF; in this case, the output is a set of formulae;
- Or the aim is to encode argumentative semantics; in this case, the output is a set of logical formulae whose models correspond to extensions or labellings for a given argumentation semantics possibly with additional constraints; sometimes, it is also possible to check whether some "candidates" are actual extensions or labellings for a given semantics.

It is worth noting that if an approach covers the first aim, it also covers at least part of the second one. Indeed, it is very difficult to encode an argumentation graph without taking into account the meaning attached to the notion of attack (at least the notion of conflict-freeness). Table 2 synthetizes the aims of the reviewed approaches.

(3) The third criterion is about *logic*. Several logics are encountered: propositional logic, first-order logic, QBF formalism, logic programming, modal logic, intuitionistic logic. Table 3 synthetizes the different cases.

Table 3
The third comparison criterion: the used logic

Used logics	Approaches
Propositional	[11,12,20,21,43,52,56,59,67]
First-order (with finite domains)	[25,30,34,36,55,93]
QBF	[4,5,49,50]
Logic programming	[24,26,54,65,78,83,84,89]
Modal	[25,70,93]
Intuitionistic	[63]

Table 4
The fourth comparison criterion: the implementations

Some existing implementations	Approaches
None	[25,43,55,63,67,70,93]
DIAMOND, UNREAL, GrappaVis	[20,21]
GRAFIX	[30,34]
SESAME, CoQuiAAS	[11–13,72,74,95]
CEGARTIX	[59]
QADF	[20,21,49,50]
PrefSAT, ArgSemSAT	[36,37]
LabSAT	[9,9]
ASPARTIX, ASPeRiX, ASP-solvers	[24,26,46,54,61,69,83,84,88,89]

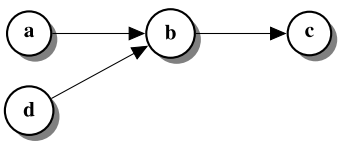
Note that the complexity of the chosen logic must be taken into account: it is more or less difficult to compute models and to establish the link between these models and the output that must be produced. This point is related to the next criterion.

(4) The fourth criterion is about the existence of *implementations*: some approaches are yet to be implemented, whereas some others have led to different implementations, some of them being used in the ICCMA competition (see for instance [72]). Table 4 synthetizes all these implementations.

12.2. An illustrating example

The following example illustrates the different encodings that can be obtained.

Ex. 2.1.1 (cont'd) The AF is a simple Dung's framework that has only one complete extension: $\{a, d, c\}$.



With ADF approach (translation of the AF into a dependence graph): See Section 3. The AF is encoded by the following dependence formulae:

$$\varphi_a = \top, \quad \varphi_b = \neg a \wedge \neg d, \quad \varphi_c = \neg b, \quad \varphi_d = \top.$$

The language is propositional with the vocabulary $V = \{a, b, c, d\}$. Then, using the ADF machinery (the computation of the fixed points of the Γ_D operator) a 3-valued model is produced, which corresponds to the complete extension $\{a, d, c\}$.

Translation of the AF into a logical base: Several approaches exist, each of them including an encoding of semantics.

[30,33,34]: See Section 4.2. The vocabulary is defined as follows:

$$V = \{Acc(a), NAcc(a), Acc(b), NAcc(b), Acc(c), NAcc(c), Acc(d), NAcc(d)\}.$$

Note that it is a first-order approach whose term models are finite so it is equivalent to a propositional approach.
The AF is encoded by the set of formulae:

$$\begin{aligned}\Sigma = \{&Acc(a) \to NAcc(b), \quad Acc(d) \to NAcc(b),\\ &Acc(b) \to NAcc(c),\\ &NAcc(a) \to \neg Acc(a), \quad NAcc(b) \to \neg Acc(b),\\ &NAcc(c) \to \neg Acc(c), \quad NAcc(d) \to \neg Acc(d)\}.\end{aligned}$$

The different requirements of the standard semantics are also logically encoded. For instance, for the complete semantics, the defence and the reinstatement principles are respectively encoded by:

$$\Sigma_d = \{\neg Acc(b), \quad Acc(c) \to (Acc(a) \vee Acc(d))\}$$
$$\Sigma_r = \{Acc(a), \quad Acc(d), \quad Acc(a) \to Acc(c), \quad Acc(d) \to Acc(c)\}.$$

Then, the models of $\Sigma \cup \Sigma_d \cup \Sigma_r$ characterize the complete extensions (here there is only one model corresponding to the unique complete extension $\{a, d, c\}$).

[67]: See Section 4.3. The language is propositional with the vocabulary $V = \{a, b, c, d, Na, Nb, Nc, Nd\}$. The AF and some requirements corresponding to the semantics are logically encoded. However, the logical encoding mixes the part issued from the input AF and the part issued from the encoding of semantics. Moreover, it is not possible to identify the part issued from each principle in the logical encoding of a semantics. For the considered AF, the resulting base is:[31]

$$\begin{aligned}\{&Na \to \neg a, \quad Nb \to \neg b, \quad Nc \to \neg c, \quad Nd \to \neg d,\\ &a, \quad d, \quad a \to Nb, \quad d \to Nb, \quad b \to Nc,\\ &a \leftrightarrow \top, \quad d \leftrightarrow \top, \quad b \leftrightarrow (Na \wedge Nd), \quad c \leftrightarrow Nb,\\ &(\neg a \wedge \neg d \wedge (\neg Na \vee \neg Nd)) \to (\neg b \wedge \neg Nb),\\ &(\neg b \wedge \neg Nb) \to (\neg c \wedge \neg Nc)\}.\end{aligned}$$

[31] Without trivial formulae such as, for instance, $(\top \wedge \bot) \to (\neg a \wedge \neg Na)$.

The models of this base characterize the complete extensions (here there is only one model corresponding to the unique complete extension $\{a, d, c\}$).

[55]: See Section 6.1. The language is first-order. Considering that the input AF is also the universe, the encoding produces the formula Φ:

$$\big(on\big(\{a, b, c, d\}\big)\big) \wedge \big(\{a\} \rhd \{b\}\big) \wedge \big(\{d\} \rhd \{b\}\big) \wedge \big(\{b\} \rhd \{c\}\big).$$

Then, considering a set of arguments t, one can check whether t is an extension for the given semantics. For instance, t is a complete extension if the following formula is a tautology:

$$\Phi \rightarrow \begin{pmatrix} on(t) \\ \wedge \neg(t \rhd t) \\ \wedge (t \rhd\!\!\rhd t) \\ \wedge \forall t_2((singl(t_2) \wedge (t \rhd\!\!\rhd t_2)) \rightarrow (t_2 \subseteq t)) \end{pmatrix}$$

[26,83,84]: See Section 9.1. The approach uses logic programming. The AF is encoded by the following logic program:

$$def(b) \leftarrow not\ def(a)$$
$$def(b) \leftarrow not\ def(d)$$
$$def(c) \leftarrow not\ def(b)$$
$$def(c) \leftarrow def(a), def(d)$$
$$def(b) \leftarrow \top$$
$$acc(a) \leftarrow not\ def(a)$$
$$acc(b) \leftarrow not\ def(b)$$
$$acc(c) \leftarrow not\ def(c)$$
$$acc(d) \leftarrow not\ def(d)$$

The complete extensions are characterized by the supported models of the logic program. Here, there is only one supported model ($\{acc(a), acc(c), acc(d), def(b)\}$) corresponding to the unique complete extension ($\{a, c, d\}$).

[24]: See Section 9.1. The approach also uses logic programming, with propositional symbols. The AF is encoded by the following logic program:

$$b \leftarrow not\ a, not\ d$$
$$c \leftarrow not\ b$$
$$a \leftarrow$$
$$d \leftarrow$$

There are correspondences between 3-valued models of the logic program and argumentation semantics. For instance, the complete extensions of the AF are characterized by the 3-valued stable models of the associated logic program.

[63]: See Section 11. The logic is equilibrium logic, with the vocabulary $V = \{a, b, c, d\}$. The AF is encoded by the base:

$$a$$
$$b$$
$$c$$
$$d$$
$$a \rightsquigarrow b$$
$$d \rightsquigarrow b$$
$$b \rightsquigarrow c$$

However, only the stable semantics has been characterized in terms of equilibrium models.

Encoding of semantics The approaches presented above allow the characterization of some semantics, given the logical encoding of the AF. Some other approaches give a characterization of semantics without encoding the input AF. They try to identify and to encode some principles governing the input semantics. In some cases, the produced formulae can be instantiated on a given AF, particularly in order to compute the extensions/labellings, or to check whether a given set of arguments is an extension (or whether a given labelling is correct w.r.t. the input semantics).

[11]: See Section 7.1. The language is propositional with the vocabulary $V = \{a, b, c, d\}$. For the considered AF, the complete extensions are characterized by the following formula Φ:

$$[(a \rightarrow \neg b) \wedge (a \leftrightarrow \top)] \wedge$$
$$[(d \rightarrow \neg b) \wedge (d \leftrightarrow \top)] \wedge$$
$$[(b \rightarrow \neg c) \wedge (b \leftrightarrow \bot)] \wedge$$
$$[(c \rightarrow \top) \wedge (c \leftrightarrow (a \vee d))]$$

Then, given a set of arguments S, it can be checked whether S is a complete extension.

[59]: See Section 7.2. The language is propositional with the vocabulary $V = \{x_a, y_a, x_b, y_b, x_c, y_c, x_d, y_d\}$. For the considered AF, the complete extensions are characterized by the following formula Φ:

$$[(\neg x_a \vee \neg x_b) \wedge (\neg x_d \vee \neg x_b) \wedge (\neg x_b \vee \neg x_c)] \wedge$$
$$[(\neg x_b) \wedge (x_c \rightarrow (x_a \vee x_d))] \wedge$$
$$[(y_a \leftrightarrow x_a) \wedge (y_d \leftrightarrow x_d)] \wedge$$
$$[y_b \leftrightarrow (x_b \vee x_a \vee x_d)] \wedge$$
$$[y_c \leftrightarrow (x_c \vee x_b)] \wedge$$
$$[y_a \wedge y_d \wedge ((y_a \wedge y_d) \rightarrow y_b)] \wedge$$
$$[y_b \rightarrow y_c]$$

Models of Φ characterize the complete extensions of the AF (x_i is *true* in the model iff the argument i belongs to the extension).

[4,5]: See Section 5. The logic uses quantified Boolean formulae (QBF formalism). The formulae can be instantiated on a given AF. For the considered AF, the complete labellings are characterized by the following formula:

$$
\begin{aligned}
&\big[\mathtt{val}(a,t) \wedge \neg\mathtt{val}(a,f) \wedge \neg\mathtt{val}(a,u)\big]\wedge\\
&\big[\mathtt{val}(d,t) \wedge \neg\mathtt{val}(d,f) \wedge \neg\mathtt{val}(d,u)\big]\wedge\\
&\big[\mathtt{val}(b,t) \rightarrow \big(\mathtt{val}(a,f) \wedge \mathtt{val}(d,f)\big)\big]\wedge\\
&\big[\mathtt{val}(b,f) \rightarrow \big(\mathtt{val}(a,t) \vee \mathtt{val}(d,t)\big)\big]\wedge\\
&\left[\mathtt{val}(b,u) \rightarrow \begin{pmatrix} \neg(\mathtt{val}(a,f) \wedge \mathtt{val}(d,f)) \\ \wedge(\neg(\mathtt{val}(a,t) \vee \mathtt{val}(d,t))) \end{pmatrix}\right]\wedge\\
&\big[\mathtt{val}(c,t) \rightarrow \mathtt{val}(b,f)\big]\wedge\\
&\big[\mathtt{val}(c,f) \rightarrow \mathtt{val}(b,t)\big]\wedge\\
&\big[\mathtt{val}(c,u) \rightarrow \big(\neg\mathtt{val}(b,f) \wedge \neg\mathtt{val}(b,t)\big)\big]\wedge\\
&\big[\neg\big(a^{\oplus} \wedge a^{\ominus}\big)\big]\wedge\\
&\big[\neg\big(b^{\oplus} \wedge b^{\ominus}\big)\big]\wedge\\
&\big[\neg\big(c^{\oplus} \wedge c^{\ominus}\big)\big]\wedge\\
&\big[\neg\big(d^{\oplus} \wedge d^{\ominus}\big)\big]
\end{aligned}
$$

[25] with first-order logic: See Section 8.1. For the considered AF, the complete labellings are characterized by the following formula:

$$
\begin{aligned}
&\big[Q_0(a) \vee Q_1(a) \vee Q_?(a)\big]\wedge\\
&\big[Q_0(b) \vee Q_1(b) \vee Q_?(b)\big]\wedge\\
&\big[Q_0(c) \vee Q_1(c) \vee Q_?(c)\big]\wedge\\
&\big[Q_0(d) \vee Q_1(d) \vee Q_?(d)\big]\wedge\\
&\big[\neg\big(Q_0(a) \wedge Q_1(a)\big) \wedge \neg\big(Q_0(a) \wedge Q_?(a)\big) \wedge \neg\big(Q_?(a) \wedge Q_1(a)\big)\big]\wedge\\
&\big[\neg\big(Q_0(b) \wedge Q_1(b)\big) \wedge \neg\big(Q_0(b) \wedge Q_?(b)\big) \wedge \neg\big(Q_?(b) \wedge Q_1(b)\big)\big]\wedge\\
&\big[\neg\big(Q_0(c) \wedge Q_1(c)\big) \wedge \neg\big(Q_0(c) \wedge Q_?(c)\big) \wedge \neg\big(Q_?(c) \wedge Q_1(c)\big)\big]\wedge\\
&\big[\neg\big(Q_0(d) \wedge Q_1(d)\big) \wedge \neg\big(Q_0(d) \wedge Q_?(d)\big) \wedge \neg\big(Q_?(d) \wedge Q_1(d)\big)\big]\wedge\\
&\big[\big(Q_0(a) \rightarrow Q_1(b)\big) \wedge \big(Q_0(d) \rightarrow Q_1(b)\big) \wedge Q_1(a) \wedge Q_1(d)\big]\wedge\\
&\big[Q_0(b) \rightarrow Q_1(c)\big]\wedge\\
&\big[\big(\big(Q_1(a) \vee Q_1(d)\big) \rightarrow Q_0(b)\big) \wedge \big(Q_1(b) \rightarrow Q_0(c)\big)\big]\wedge
\end{aligned}
$$

$$\big[\big(\big(Q_0(b) \vee Q_?(b)\big) \wedge Q_?(b)\big) \rightarrow Q_?(c)\big] \wedge$$
$$\left[\begin{pmatrix} (Q_0(a) \vee Q_?(a)) \\ \wedge(Q_0(d) \vee Q_?(d)) \\ \wedge(Q_?(a) \vee Q_?(d)) \end{pmatrix} \rightarrow Q_?(b)\right]$$

[25] with modal logic: See Section 10.3. The modal formula characterizing the complete extensions is quite simple. The difficulty lies in computational issues.

$$E \text{ is a complete extension iff } E = \Box(\neg E \wedge \Diamond E)$$

[70]: See Section 10.2. The logic is a modal logic. The modal formula characterizing the complete extensions is the following one:

$$[\forall]\begin{pmatrix} (1 \leftrightarrow [\leftarrow]0) \\ \wedge(0 \leftrightarrow \langle\leftarrow\rangle 1) \\ \wedge[(1 \wedge \neg 0 \wedge \neg ?) \vee (\neg 1 \wedge 0 \wedge \neg ?) \vee (\neg 1 \wedge \neg 0 \wedge ?)] \end{pmatrix}$$

Here again, the difficulty lies in computational issues.

12.3. Conclusion

Considering that this survey is published on the occasion of the 25 years of Dung's approach, we have chosen to discuss approaches that relate logical theories and abstract argumentation in a specific sense: "How logic has been used for capturing various aspects or parts of Dung's argumentation". In view of the numerous proposals on the topic, it seemed to us of interest to look at them in the same place (this paper) and in a common way (whereas authors often introduce their approaches with some bias, no matter how sensible and well-taken). So, the presentation of each approach has been designed in order to answer the same set of questions: What is the aim? What are the "inputs" and the "outputs"? What is the type of the logic used? Do implementations exist?

In Section 12.1, we have given some criteria/traits/features/properties to get comparisons [similarities and differences] out of the answers to these questions; then in Section 12.2, a same example has been used in order to illustrate the behaviour of the presented approaches. The number and the diversity of these approaches are so huge that this comparison is limited but we still believe it has some value. Indeed, each presented approach has its own specificities, sometimes antagonistic with the other approaches, but also justified by its own context of definition or use. And we think that it is difficult and perhaps impossible to give here a more precise evaluation.

Nevertheless, some "use cases" could be proposed for which an approach makes a better choice that another one.

A first example of "use cases" could be: when your "input" is an AF (only simple attacks between arguments) and we are only interested in a very efficient computation of extensions, an encoding using [11,12] coupled with a SAT solver could be a good choice (note that some of the best solvers of the ICCMA competition do this choice [16]).

Another example of "use cases" could be: your "input" is an argumentation framework with several enrichments such as weighting functions, and we want to compute labellings. Obviously the approach

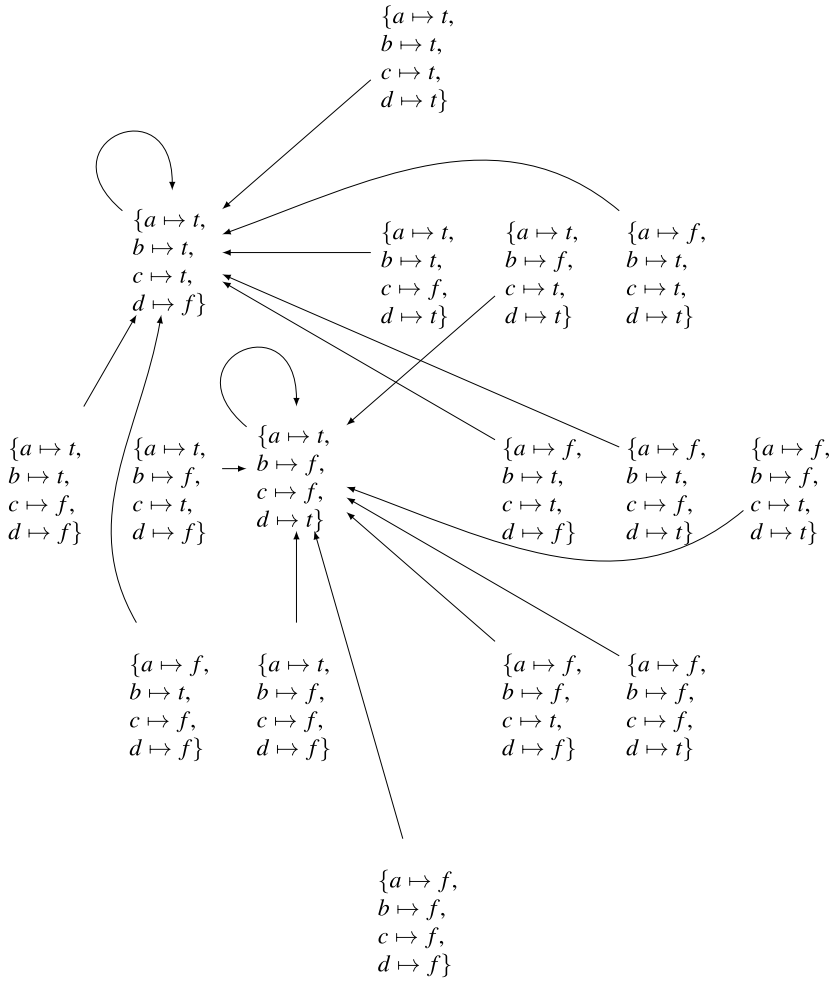


Fig. 3. ADF approach, Example 3.2.1: set of two-valued interpretations and G_D operator.

defined in [11,12] cannot be used whereas an encoding by an ADF [21] coupled with one of the ADF solvers (and there are many ADF solvers) could be a better choice.

But if your input is an argumentation framework with recursive interactions, perhaps it would be better to use an encoding of this framework using the approach proposed in [33] coupled with a SAT solver in order to obtain the extensions.

Another point of view could be the size of the logical knowledge base corresponding to the encoded framework: clearly the encoding in propositional logic proposed in [59] is lengthy compared to the encoding in modal logic proposed by [70].

On another hand, expressiveness can also be taken into account since it differs depending on the approach under consideration. An example can be found when we compare the YALLA approach for instance with the encoding proposed in [33]: both use a first-order language, but YALLA is also able to reason about extensions, and not only about argument acceptance (for each semantics, there is a first-order formula in YALLA expressing that a subset of arguments is an extension under this semantics).

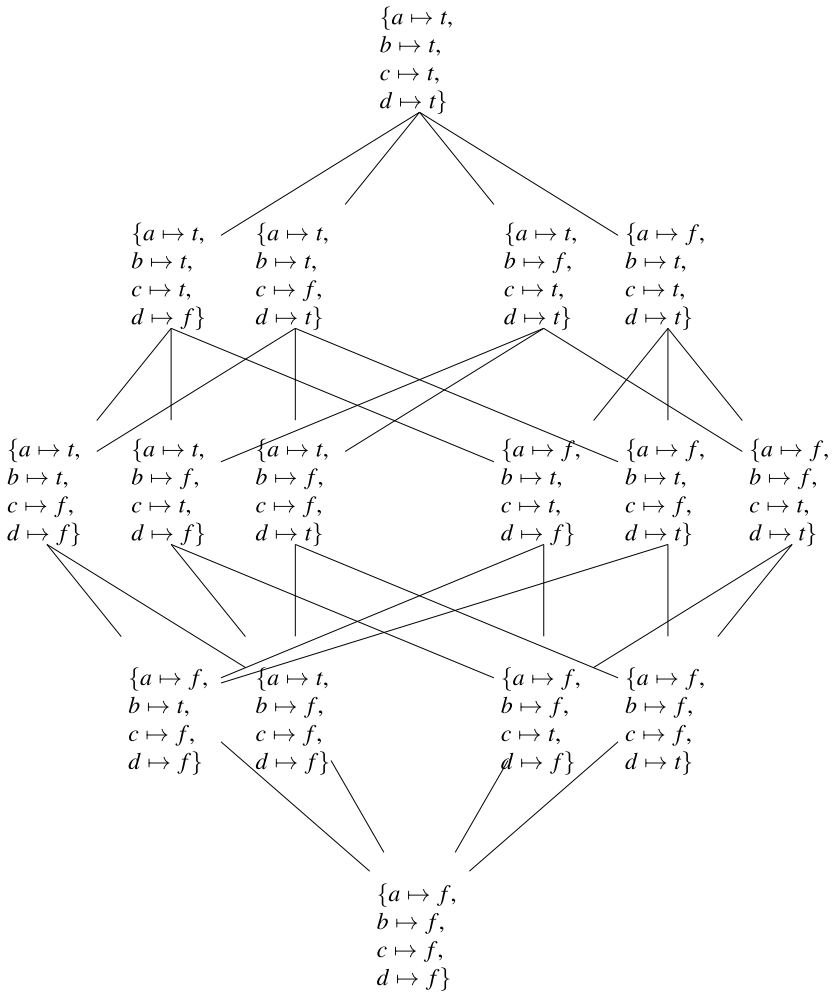


Fig. 4. ADF approach, Example 3.2.1: the complete lattice built from the set of two-valued interpretations and the $\leqslant_t$ preordering.

And of course, in a computational point of view, an existing and efficient implementation could offer a preponderant advantage. For instance, this is generally the case of approaches using propositional logic [11,59] compared to those using modal logic [25,70].

All previous "use cases" clearly show the multiplicity of points of view and that leads us to conclude that one cannot obtain a definite answer to the question of what approach is best.

Acknowledgements

This work has been supported by the Centre International de Mathématiques et d'Informatique de Toulouse (CIMI) through contract ANR-11-LABEX-0040-CIMI within the program ANR-11-IDEX-0002-02. Moreover, the authors wish to thank Sylvie Doutre and Luis Fariñas del Cerro for the fruitful exchanges and discussions.

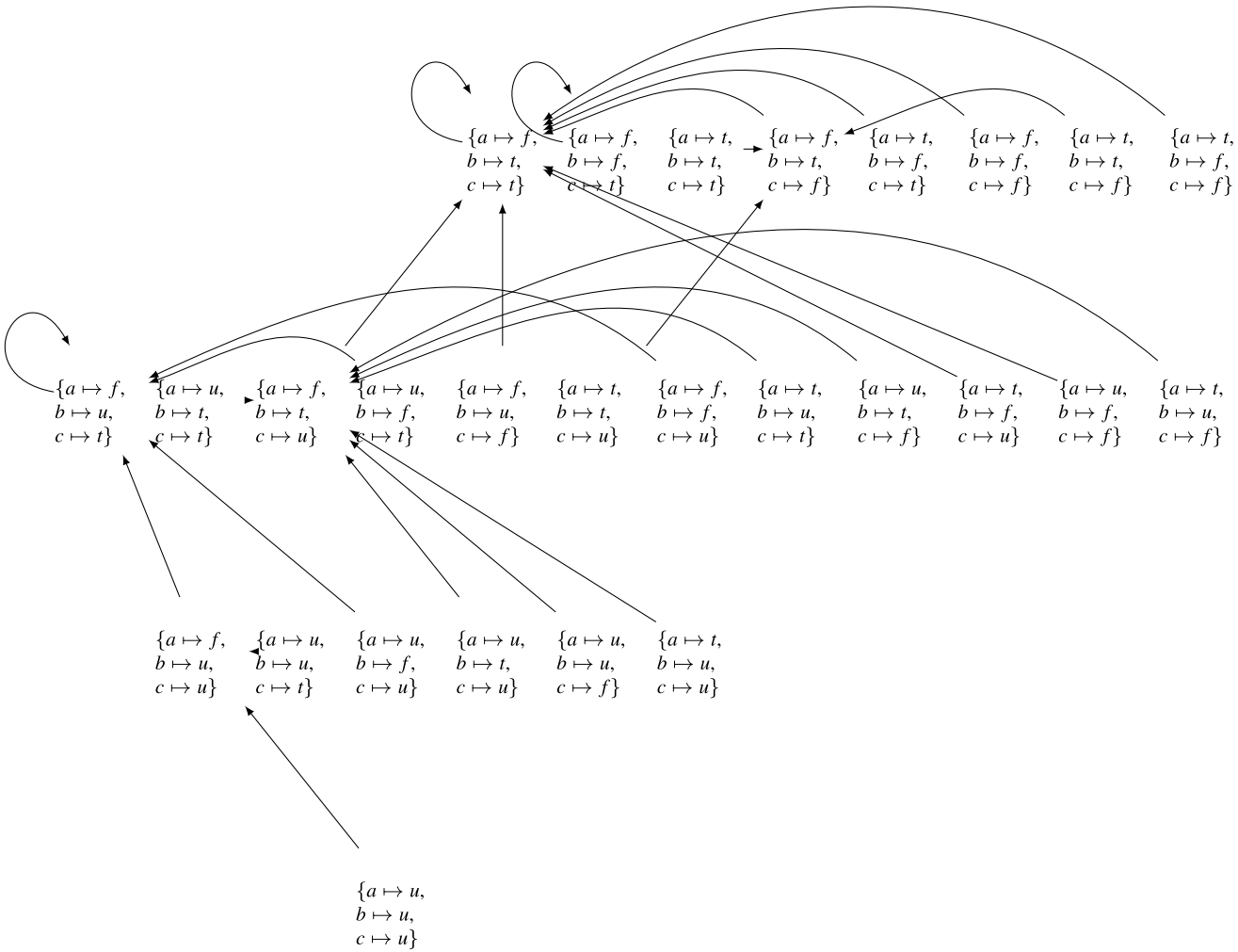


Fig. 5. ADF approach, Example 3.2.2: set of three-valued interpretations and Γ_D operator.

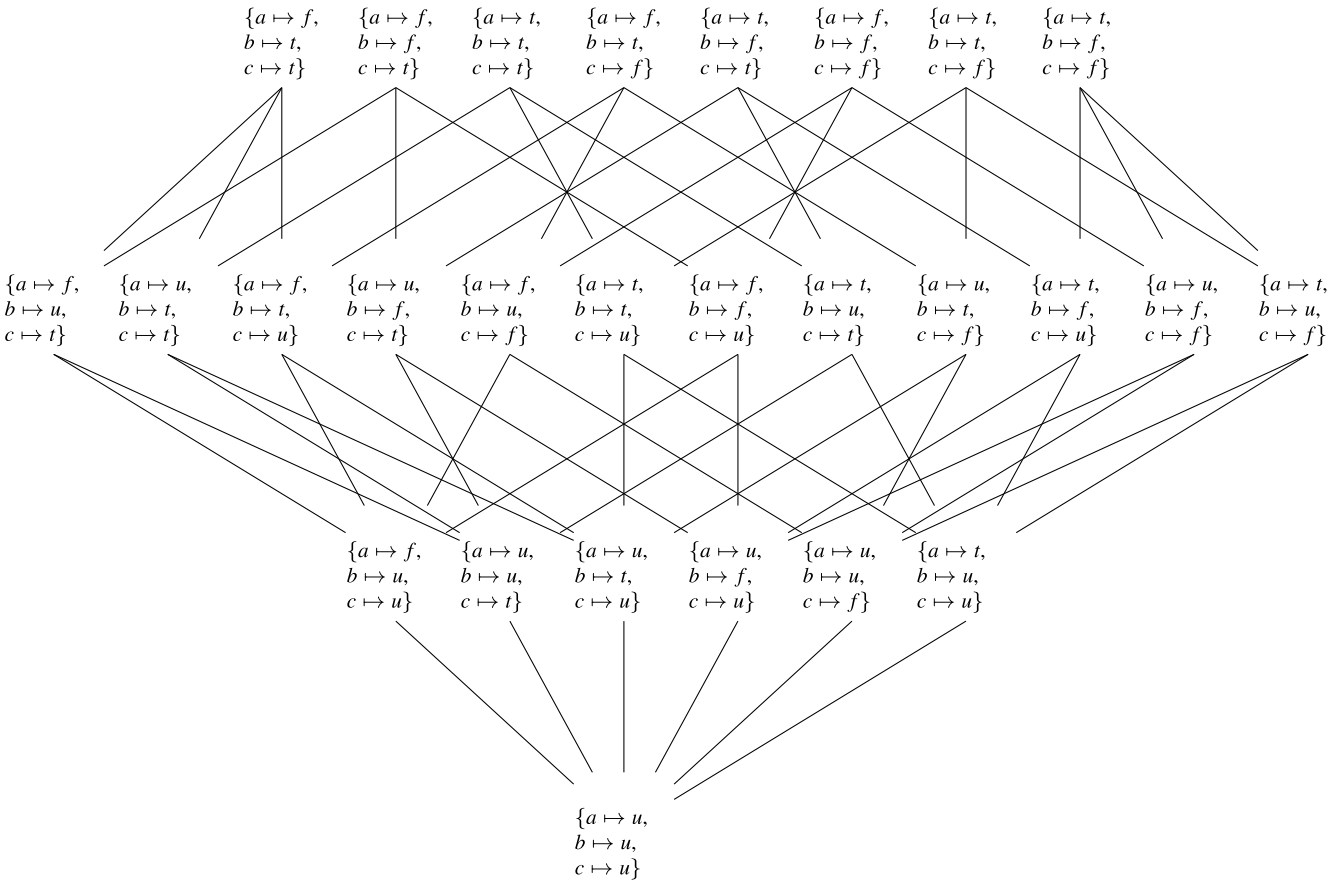


Fig. 6. ADF approach, Example 3.2.2: the complete meet-lattice built from the set of three-valued interpretations and the $\leqslant_i$ preordering.

References

[1] M. Alviano, Argumentation reasoning via circumscription with Pyglaf, *Fundamenta Informaticae* **167**(1–2) (2019), 1–30. doi:10.3233/FI-2019-1808.

[2] L. Amgoud and C. Cayrol, A reasoning model based on the production of acceptable arguments, *Annals of Mathematics and Artificial Intelligence* **34** (2002), 197–216. doi:10.1023/A:1014490210693.

[3] L. Amgoud, N. Maudet and S. Parsons, Modelling dialogues using argumentation, in: *Proc. of ICMAS*, 2000, pp. 31–38.

[4] O. Arieli and M. Caminada, A general QBF-based formalization of abstract argumentation theory, in: *Proceedings of the 4th Conference on Computational Models of Argument (COMMA'2012)*, B. Verheij, S. Szeider and S. Woltran, eds, Frontiers in Artificial Intelligence and Applications, Vol. 245, IOS Press, Vienna, Austria, 2012, pp. 105–116.

[5] O. Arieli and M.W.A. Caminada, A QBF-based formalization of abstract argumentation semantics, *J. Applied Logic* **11**(2) (2013), 229–252. doi:10.1016/j.jal.2013.03.009.

[6] P. Baroni, M. Caminada and M. Giacomin, An introduction to argumentation semantics, *Knowledge Eng. Review* **26**(4) (2011), 365–410. doi:10.1017/S0269888911000166.

[7] P. Baroni, F. Cerutti, M. Giacomin and G. Guida, AFRA: Argumentation framework with recursive attacks, *International Journal of Approximate Reasoning* **52** (2011), 19–37. doi:10.1016/j.ijar.2010.05.004.

[8] H. Barringer, D.M. Gabbay and J. Woods, Temporal dynamics of support and attack networks: From argumentation to zoology, in: *Mechanizing Mathematical Reasoning*, D. Hutter and W. Stephan, eds, LNAI, Vol. 2605, Springer-Verlag, 2005, pp. 59–98. doi:10.1007/978-3-540-32254-2_5.

[9] 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: *KI 2015: Advances in Artificial Intelligence – 38th Annual German Conference on AI, Proceedings*, Dresden, Germany, September 21–25, 2015, 2015, pp. 241–248.

[10] P. Besnard, J. Boudou, S. Doutre, Van Hieu Ho and D. Longin, The SESAME website, https://www.irit.fr/SESAME/.

[11] P. Besnard and S. Doutre, Checking the acceptability of a set of arguments, in: *10th International Workshop on Non-Monotonic Reasoning (NMR'04)*, J.P. Delgrande and T. Schaub, eds, Whistler, Canada, 2004, pp. 59–64.

[12] P. Besnard, S. Doutre and A. Herzig, Encoding argument graphs in logic (regular paper), in: *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, Montpellier, France, 15/07/2014–19/07/2014, A. Laurent, O. Strauss, B. Bouchon-Meunier and R.R. Yager, eds, Communications in Computer and Information Science, Vol. 443, Springer, http://www.springerlink.com, 2014, pp. 345–354.

[13] P. Besnard, S. Doutre, V.H. Ho and D. Longin, SESAME – a system for specifying semantics in abstract argumentation (regular paper), in: *International Workshop on Systems and Algorithms for Formal Argumentation (SAFA)*, Potsdam, Germany, 13/09/2016, M. Thimm, F. Cerutti, H. Strass and M. Vallati, eds, CEUR Workshop Proceedings, Vol. 1672, http://CEUR-WS.org, 2016, pp. 40–51.

[14] P. Besnard and A. Hunter, A logic-based theory of deductive arguments, *Artificial Intelligence* **128**(1–2) (2001), 203–235. doi:10.1016/S0004-3702(01)00071-6.

[15] P. Bisquert, C. Cayrol, F. Dupin de Saint-Cyr Bannay and M.-C. Lagasquie-Schiex, Characterizing change in abstract argumentation systems, in: *Trends in Belief Revision and Argumentation Dynamics*, E. Fermé, D. Gabbay and G. Simari, eds, Studies in Logic, Vol. 48, College Publications, http://www.collegepublications.co.uk/, 2013, pp. 75–102.

[16] S. Bistarelli, F. Santini, L. Kotthoff, T. Mantadelis and C. Taticchi, Int. competition on computational models of argumentation, 2019, https://www.iccma2019.dmi.unipg.it/.

[17] G. Boella, D.M. Gabbay, L. van der Torre and S. Villata, Support in abstract argumentation, in: *Proceeding of the 2010 Conference on Computational Models of Argument: Proceedings of COMMA 2010*, IOS Press, Amsterdam, The Netherlands, 2010, pp. 111–122.

[18] R. Booth, S. Kaci, T. Rienstra and L. van der Torre, A logical theory about dynamics in abstract argumentation, in: *Scalable Uncertainty Management*, W. Liu, V.S. Subrahmanian and J. Wijsen, eds, Lecture Notes in Computer Science, Vol. 8078, Springer, Berlin, Heidelberg, 2013, pp. 148–161. doi:10.1007/978-3-642-40381-1_12.

[19] G. Brewka, M. Diller, G. Heissenberger, T. Linsbichler and S. Woltran, Solving advanced argumentation problems with answer-set programming, in: *Proc. of AAAI*, 2017, pp. 1077–1083.

[20] G. Brewka, S. Ellmauthaler, H. Strass, J.P. Wallner and S. Woltran, Abstract Dialectical Frameworks. An overview, *The IfCoLog Journal of Logics and Their Applications* **4**(8) (2017), 2263–2318.

[21] G. Brewka, S. Ellmauthaler, H. Strass, J.P. Wallner and S. Woltran, Abstract Dialectical Frameworks, in: *Handbook of Formal Argumentation*, P. Baroni, D. Gabbay, M. Giacomin and L. van der Torre, eds, College Publications, 2018, pp. 237–286, Chapter 5.

[22] G. Brewka and S. Woltran, The ADF project website, http://www.dbai.tuwien.ac.at/research/project/adf/.

[23] G. Brewka and S. Woltran, The GRAPPA project website, http://www.dbai.tuwien.ac.at/proj/grappa/.

[24] M. Caminada, S. Sá, J. Alcântara and W. Dvorák, On the equivalence between logic programming semantics and argumentation semantics, *Int. J. Approx. Reasoning* **58** (2015), 87–111. doi:10.1016/j.ijar.2014.12.004.

[25] M.W.A. Caminada and D.M. Gabbay, A logical account of formal argumentation, *Studia Logica* **93**(2–3) (2009), 109–145. doi:10.1007/s11225-009-9218-x.
[26] J.L. Carballido, J.C. Nieves and M. Osorio, Inferring preferred extensions by pstable semantics, *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial* **13**(41) (2009), 38–53.
[27] C. Cayrol, F. Dupin de Saint-Cyr Bannay and M.-C. Lagasquie-Schiex, Change in abstract argumentation frameworks: Adding an argument, *Journal of Artificial Intelligence Research* **38** (2010), 49–84. doi:10.1613/jair.2965.
[28] C. Cayrol, J. Fandinno, L. Fariñas del Cerro and M.-C. Lagasquie-Schiex, Valid attacks in argumentation frameworks with recursive attacks, in: *Proc. of Thirteenth International Symposium on Commonsense Reasoning*, CEUR Workshop Proceedings, Vol. 2052, 2017.
[29] C. Cayrol, J. Fandinno, L. Fariñas del Cerro and M.-C. Lagasquie-Schiex, Argumentation frameworks with recursive attacks and evidence-based support, in: *Proc. of Tenth International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, F. Ferrarotti and S. Woltran, eds, LNCS, Vol. 10833, Springer-Verlag, 2018, pp. 150–169. doi:10.1007/978-3-319-90050-6_9.
[30] C. Cayrol, L. Fariñas del Cerro and M.-C. Lagasquie-Schiex, Logical encodings of interactions in an argumentation graph with recursive attacks, Technical report, RR-2017-08-FR, IRIT, 2017.
[31] C. Cayrol and M.-C. Lagasquie-Schiex, Gradual valuation for bipolar argumentation frameworks, in: *Proc. of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, L. Godo, ed., LNAI, Vol. 3571, Springer-Verlag, Barcelona, Spain, 2005, pp. 366–377.
[32] C. Cayrol and M.-C. Lagasquie-Schiex, Bipolarity in argumentation graphs: Towards a better understanding, *International Journal of Approximate Reasoning* **54**(7) (2013), 876–899. doi:10.1016/j.ijar.2013.03.001.
[33] C. Cayrol and M.-C. Lagasquie-Schiex, Logical encoding of argumentation frameworks with higher-order attacks, in: *Proc. of the 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2018.
[34] C. Cayrol and M.-C. Lagasquie-Schiex, Acceptability semantics in recursive argumentation frameworks: Logical encoding and computation, Rapport de recherche, IRIT/RR-2018-02-FR, IRIT, Université Paul Sabatier, Toulouse, 2018.
[35] C. Cayrol and M.-C. Lagasquie-Schiex, The Grafix website, http://www.irit.fr/grafix.
[36] F. Cerutti, P.E. Dunne, M. Giacomin and M. Vallati, Computing preferred extensions in abstract argumentation: A SAT-based approach, in: *Theory and Applications of Formal Argumentation, TAFA 2013*, E. Black, S. Modgil and N. Oren, eds, LNAI, Vol. 8306, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 176–193. doi:10.1007/978-3-642-54373-9_12.
[37] F. Cerutti, M. Giacomin and M. Vallati, ArgSemSAT: Solving argumentation problems using SAT, in: *Computational Models of Argument – Proceedings of COMMA*, 2014, pp. 455–456.
[38] F. Cerutti, M. Giacomin, M. Vallati and M. Zanella, An SCC recursive meta-algorithm for computing preferred labellings in abstract argumentation, in: *Proc. of KR*, 2014, pp. 42–51.
[39] F. Cerutti, M. Vallati and M. Giacomin, An efficient Java-based solver for abstract argumentation framework: jArgSemSAT, *Int. Journal of Articial Intelligence Tools* **26**(2) (2017), 1–26.
[40] F. Cerutti, M. Vallati, M. Giacomin and T. Zanetti, ArgSemSAT-2017, 2017.
[41] A. Cohen, S. Gottifredi, A.J. García and G.R. Simari, A survey of different approaches to support in argumentation systems, *The Knowledge Engineering Review* **29** (2014), 513–550. doi:10.1017/S0269888913000325.
[42] A. Cohen, S. Gottifredi, A.J. García and G.R. Simari, An approach to abstract argumentation with recursive attack and support, *Journal of Applied Logic* **13**(4) (2015), 509–533. doi:10.1016/j.jal.2014.12.001.
[43] S. Coste-Marquis, C. Devred and P. Marquis, Constrained argumentation frameworks, in: *Proc. of KR*, Lake District, 2006, pp. 112–122.
[44] S. Coste-Marquis, S. Konieczny, J.-G. Mailly and P. Marquis, On the revision of argumentation systems: Minimal change of argument statuses, in: *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, C. Baral and G. De Giacomo, eds, AAAI Press, 2014, pp. 72–81.
[45] S. Coste-Marquis, S. Konieczny, J.-G. Mailly and P. Marquis, A translation-based approach for revision of argumentation frameworks, in: *Logics in Artificial Intelligence*, Springer, 2014, pp. 397–411. doi:10.1007/978-3-319-11558-0_28.
[46] C. Devred, S. Doutre, C. Lefèvre and P. Nicolas, Dialectical proofs for constrained argumentation, in: *Computational Models of Argument – Proceedings of COMMA*, 2010, pp. 159–170.
[47] M. Diller, W. Dvorak, J. Puhrer, J. Wallner and S. Woltran, Applications of ASP in formal argumentation, in: *Proc. of TAASP Workshop*, 2018.
[48] M. Diller, A. Haret, T. Linsbichler, S. Rümmele and S. Woltran, An extension-based approach to belief revision in abstract argumentation, in: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 2015, pp. 2926–2932.
[49] M. Diller, J. Wallner and S. Woltran, Reasoning in Abstract Dialectical Frameworks using quantified Boolean formulas, *Argument & Computation* **6**(2) (2015), 149–177. doi:10.1080/19462166.2015.1036922.
[50] M. Diller, J.P. Wallner and S. Woltran, Reasoning in Abstract Dialectical Frameworks using quantified Boolean formulas, in: *Proceedings of the 5th Conference on Computational Models of Argument (COMMA'2014)*, S. Parsons, N. Oren,

C. Reed and F. Cerutti, eds, Frontiers in Artificial Intelligence and Applications, Vol. 266, Pitlochry, Scottish Highlands, UK, 2014, pp. 241–252, Held in Atholl Palace Hotel.

[51] Y. Dimopoulos, J.-G. Mailly and P. Moraitis, Control argumentation frameworks, in: *Proc. of AAAI*, 2018, pp. 4678–4685.

[52] S. Doutre, A. Herzig and L. Perrussel, A dynamic logic framework for abstract argumentation, in: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'2014)*, C. Baral, G. De Giacomo and T. Eiter, eds, AAAI Press, Vienna, Austria, 2014.

[53] S. Doutre and J.-G. Mailly, Constraints and changes a survey of abstract argumentation dynamics, *Argument & Computation* **9**(3) (2018), 223–248. doi:10.3233/AAC-180425.

[54] P.M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, *Artif. Intell.* **77**(2) (1995), 321–358. doi:10.1016/0004-3702(94)00041-X.

[55] F. Dupin de Saint-Cyr, P. Bisquert, C. Cayrol and M. Lagasquie-Schiex, Argumentation update in YALLA (Yet Another Logic Language for Argumentation), *Int. J. Approx. Reasoning* **75** (2016), 57–92. doi:10.1016/j.ijar.2016.04.003.

[56] W. Dvořák, S. Gaggl, T. Linsbichler and J. Wallner, Reduction-based approaches to implement Modgil's extended argumentation frameworks, in: *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation – Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, T. Eiter, H. Strass, M. Truszczynski and S. Woltran, eds, Lecture Notes in Computer Science, Vol. 9060, Springer, 2015, pp. 249–264. doi:10.1007/978-3-319-14726-0_17.

[57] W. Dvořák, S.A. Gaggl and S. Woltran, The ASPARTIX website, https://www.dbai.tuwien.ac.at/research/argumentation/aspartix/.

[58] W. Dvorak, M. Järvisalo, J. Wallner and S. Woltran, Complexity-sensitive decision procedures for abstract argumentation, *Artificial Intelligence* **206** (2014), 53–78. doi:10.1016/j.artint.2013.10.001.

[59] W. Dvořák, M. Järvisalo, J.P. Wallner and S. Woltran, CEGARTIX: A SAT-based argumentation system, in: *Proc. of Pragmatics of SAT, Workshop of SAT Conference*, 2012.

[60] W. Dvořák, M. Järvisalo, J.P. Wallner and S. Woltran, The CEGARTIX website, https://www.dbai.tuwien.ac.at/proj/argumentation/cegartix/.

[61] U. Egly, S.A. Gaggl and S. Woltran, Answer-set programming encodings for argumentation frameworks, *Argument & Computation* **1**(2) (2010), 147–177. doi:10.1080/19462166.2010.486479.

[62] W. Faber, M. Vallati, F. Cerutti and M. Giacomin, Enumerating preferred extensions using ASP domain heuristics: The ASPrMin solver, in: *Proc. of COMMA*, 2018, pp. 459–460.

[63] J. Fandinno and L. Fariñas del Cerro, Constructive logic covers argumentation and logic programming, in: *Proc. of Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2018)*, F.T.M. Thielscher and F. Wolter, eds, AAAI Press, 2018, pp. 128–137.

[64] D.M. Gabbay, Fibring argumentation frames, *Studia Logica* **93** (2009), 231–295. doi:10.1007/s11225-009-9217-y.

[65] D.M. Gabbay, Semantics for higher level attacks in extended argumentation frames part 1: Overview, *Studia Logica* **93**(2–3) (2009), 357–381. doi:10.1007/s11225-009-9211-4.

[66] D.M. Gabbay, *Meta-Logical Investigations in Argumentation Networks*, Studies in Logic, Vol. 44, College Publications, 2013.

[67] D.M. Gabbay and M. Gabbay, The attack as strong negation, part I, *Logic Journal of the IGPL* **23**(6) (2015), 881–941. doi:10.1093/jigpal/jzv033.

[68] D.M. Gabbay and M. Gabbay, The attack as intuitionistic negation, *Logic Journal of the IGPL* **24**(5) (2016), 807–837. doi:10.1093/jigpal/jzw012.

[69] S.A. Gaggl, Towards a general argumentation system based on answer-set programming, in: *Technical Communications of the 26th International Conference on Logic Programming (ICLP 2010)*, M.V. Hermenegildo and T. Schaub, eds, LIPIcs, Vol. 7, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2010, pp. 265–269.

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

[71] N. Karacapilidis and D. Papadias, Computer supported argumentation and collaborative decision making: The HERMES system, *Information Systems* **26**(4) (2001), 259–277. doi:10.1016/S0306-4379(01)00020-5.

[72] J. Lagniez, E. Lonca and J. Mailly, CoQuiAAS: A constraint-based quick abstract argumentation solver, in: *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI*, 2015, pp. 928–935.

[73] C. Lefèvre and P. Nicolas, The ASPeRiX website, http://www.info.univ-angers.fr/pub/claire/asperix/Argumentation/.

[74] T. Lehtonen, A. Niskanen and M. Järvisalo, SAT-based approaches to adjusting, repairing, and computing largest extensions of argumentation frameworks, in: *Computational Models of Argument – Proceedings of COMMA*, 2018, pp. 193–204.

[75] T. Linsbichler, M. Maratea, A. Niskanen, J. Wallner and S. Woltran, Novel algorithms for Abstract Dialectical Frameworks based on complexity analysis of subclasses and SAT solving, in: *Proc. of IJCAI*, 2018, pp. 1905–1911.

[76] S. Modgil, Reasoning about preferences in argumentation frameworks, *Artificial Intelligence* **173** (2009), 901–934. doi:10.1016/j.artint.2009.02.001.

[77] A. Niskanen and M. Järvisal, μ-toksia. Participating in ICCMA 2019, 2019.
[78] F. Nouioua, AFs with necessities: Further semantics and labelling characterization, in: *Proc. of the International Conference on Scalable Uncertainty Management (SUM)*, 2013, pp. 120–133. doi:10.1007/978-3-642-40381-1_10.
[79] F. Nouioua and V. Risch, Bipolar argumentation frameworks with specialized supports, in: *Proc. of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE Computer Society, 2010, pp. 215–218.
[80] F. Nouioua and V. Risch, Argumentation frameworks with necessities, in: *Proc. of the International Conference on Scalable Uncertainty Management (SUM)*, Springer-Verlag, 2011, pp. 163–176. doi:10.1007/978-3-642-23963-2_14.
[81] N. Oren and T.J. Norman, Semantics for evidence-based argumentation, in: *Proc of the 2nd International Conference on Computational Models of Argument (COMMA)*, P. Besnard, S. Doutre and A. Hunter, eds, IOS Press, 2008, pp. 276–284.
[82] N. Oren, C. Reed and M. Luck, Moving between argumentation frameworks, in: *Proceeding of the Conference on Computational Models of Argument (COMMA)*, IOS Press, Amsterdam, The Netherlands, 2010, pp. 379–390.
[83] M. Osorio and J.C. Nieves, Range-based argumentation semantics as two-valued models, *TPLP* **17**(1) (2017), 75–90.
[84] M. Osorio, J.C. Nieves and A. Santoyo, Complete extensions as Clark's completion semantics, in: *Mexican International Conference on Computer Science, ENC 2013*, Morelia, Michoacán, Mexico, October 30–Nov. 1, 2013, 2013, pp. 81–88.
[85] S. Polberg, Understanding the abstract dialectical framework, in: *Proc. of JELIA*, 2016, pp. 430–446.
[86] S. Polberg, Intertranslatability of abstract argumentation frameworks, Technical report, DBAI-TR-2017-104, DBAI (Institut für Informationssysteme Abteilung Datenbanken und Artificial Intelligence Technische Universität Wien), 2017.
[87] S. Polberg and N. Oren, Revisiting support in abstract argumentation systems, Technical report, TU Wien, Institut for Informatics, 2014.
[88] F. Pu, G. Luo and Z. Jiang, Encoding argumentation semantics by Boolean algebra, *IEICE Transactions* **100D**(4) (2017), 838–848. doi:10.1587/transinf.2016EDP7313.
[89] C. Sakama and T. Rienstra, Representing argumentation frameworks in answer set programming, *Fundamenta Informaticae* **155**(3) (2017), 261–292. doi:10.3233/FI-2017-1585.
[90] F. Toni and M. Sergot, Argumentation and answer set programming, in: *Logic Programming, Knowledge Representation and Nonmonotonic Reasoning: Essays in Honor of Michael Gelfond*, M. Balduccini and T.C. Son, eds, Springer, 2010, pp. 164–180.
[91] B. Verheij, DEFLOG: On the logical interpretation of prima facie justified assumptions, *Journal of Logic in Computation* **13** (2003), 319–346. doi:10.1093/logcom/13.3.319.
[92] S. Villata, G. Boella, D.M. Gabbay and L. van der Torre, Modelling defeasible and prioritized support in bipolar argumentation, *Annals of Mathematics and Artificial Intelligence* **66**(1–4) (2012), 163–197. doi:10.1007/s10472-012-9317-7.
[93] S. Villata, G. Boella, D.M. Gabbay, L. van der Torre and J. Hulstijn, A logic of argumentation for specification and verification of abstract argumentation frameworks, *Annals of Mathematics and Artificial Intelligence* **66**(1–4) (2012), 199–230. doi:10.1007/s10472-012-9318-6.
[94] M. Walicki and S. Dyrkolbotn, Finding kernels or solving SAT, *Discrete Algorithms* **10** (2012), 146–164. doi:10.1016/j.jda.2011.11.004.
[95] J.P. Wallner, G. Weissenbacher and S. Woltran, Advanced SAT techniques for abstract argumentation, in: *Computational Logic in Multi-Agent Systems – 14th International Workshop, CLIMA XIV, Proceedings*, Corunna, Spain, September 16–18, 2013, 2013, pp. 138–154.
[96] Y. Wu, M. Caminada and D.M. Gabbay, Complete extensions in argumentation coincide with 3-valued stable models in logic programming, *Studia Logica: An International Journal for Symbolic Logic* **93**(2/3) (2009), 383–403. doi:10.1007/s11225-009-9210-5.