Logical Frameworks and Abstract Argumentation: A Survey of existing works

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

Tech. Report
IRIT/RR- -2019- -01- -FR

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.

Abstract

This work presents a survey of existing works about “Logics and abstract 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.

Contents

1 Introduction

The relationship between abstract argumentation and logic has actually been exploited right from the start, in the seminal article [Dun95] introducing abstract argumentation where Dung establishes a formal equivalence between argumentation frameworks and logic programs. While one could expect this to be based on the fact that argumentation involves reasoning of some kind, the relationship at work happens to be different. Instead, a great deal of papers have been devoted to defining correspondences that permit to use logic to obtain various results in argumentation. These results can be meta-level, namely properties of argumentation frameworks, or object-level, for instance, specific extensions of a given argumentation framework. The other way around, there has also been some work on translating and interpreting logic programs in terms of abstract argumentation.

This article is a survey about abstract argumentation papers that have a strong emphasis on logic. As just mentioned, two categories can be defined depending on how whether such a work (1) is an approach taking advantage of logic to capture various aspects of abstract argumentation or (2) is an approach embedding structures such as logic programs into abstract argumentation. Again, the former category gathers a lot of papers whereas the latter category is much more limited, mostly referring to works which give an argumentative meaning to logic programs.

Focussing on the many papers that in fact consist of 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:

- 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. Then, as already mentioned, the category "embedding logic structures into argumentation" gathers few works as compared to the other category, and, for this reason, these works are all reported in a single section, namely Section 9.3. The many works in the other category are split over several sections, as follows.

Section 3 is devoted to a single approach, Abstract Dialectical Frameworks, 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. 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 (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. 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, 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. 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 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 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. Some of these works encode an argumentation graph into a logic program. Other works encode a normal logic program into an argumentation graph. The section describes different mappings which allow to transform an argumentation graph into a logic program and vice-versa, 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, 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.

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

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. 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.

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

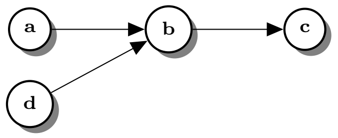
2.1 Argumentation frameworks

According to [Dun95], an abstract argumentation framework consists of a set of arguments together with a binary relation between arguments.

Def. 1 (AF [Dun95])**.** *An* argumentation framework *(AF) is a pair* $(\mathcal{A}, \mathcal{R})$ *where* $\mathcal{A}$ *is a set*[1] *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,[2] 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):*

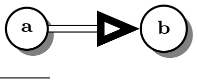


Many extensions of this framework have been proposed. For instance, bipolar abstract frameworks have been introduced first in [KP01, Ver03]. They include a second relation between arguments, the support relation, that is a positive interaction (in contrast to the attack relation that is a negative one). In [CLS05], the support relation is left general so that the bipolar framework keeps a high level of abstraction.

Def. 2 (BAF [CLS05])**.** *A* bipolar argumentation framework *(BAF) is a triple* $(\mathcal{A}, \mathcal{R}_{att}, \mathcal{R}_{sup})$ *where* $\mathcal{A}$ *is a set of abstract arguments,* $\mathcal{R}_{att} \subseteq \mathcal{A} \times \mathcal{A}$ *(resp.* $\mathcal{R}_{sup} \subseteq \mathcal{A} \times \mathcal{A}$*) is a binary relation on* $\mathcal{A}$*, called the attack (resp. support) relation.*

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.*



[1] Generally assumed to be finite.

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

However, there is no single interpretation of the support, and a number of researchers proposed specialized variants of the support relation (deductive support [BGvdTV10], necessary support [NR10, NR11], evidential support [ON08, ORL10]). These proposals have been developed quite independently, based on different intuitions and with different formalizations. [CLS13] presents a comparative study in order to restate these proposals in a common setting, the bipolar argumentation framework (see also [CGGS14] 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 [CFFLS18]:[3]

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 [BGW05] 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 [Mod09]), or for modelling situations where an attack might be defeated by an argument, without contesting the acceptability of the source of the attack [BCGG11]. Attacks to attacks and supports have been first considered in [Gab09a] with higher level networks, then in [VBGvdT12]; and more generally, [CGGS15] proposes an Attack-Support Argumentation Framework which allows for nested attacks and supports, i.e. attacks and supports whose targets can be other attacks or supports, at any level. 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.

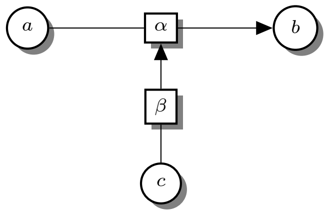
Def. 4 (RAF [CFFLS17])**.** *A* recursive argumentation framework (RAF) *is a tuple* $\langle \mathcal{A}, \mathcal{R}, \mathbf{s}, \mathbf{t} \rangle$ *where* $\mathcal{A}$ *is a finite and non-empty set of arguments,* $\mathcal{R}$ *is a finite set disjunct from* $\mathcal{A}$ *representing attack 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.

[3]The first definition of EBAF was given in [ON08] then modified in [PO14].

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:*



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

Still other extensions exist such as frameworks with collective interactions (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 [Dun95] 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 [Dun95])**.** *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 *wrt 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 wrt $S\}$.*
- *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 (wrt $\subseteq$) fixed point*[4] *of $\mathcal{F}$.*

- *S is* a preferred extension *of $(\mathcal{A}, \mathcal{R})$ iff it is a maximal (wrt $\subseteq$) 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.*

Acceptability semantics can also be defined in terms of labellings, as in [BCG11], for instance.

Def. 6 (Labelling [BCG11])**.** *Let $(\mathcal{A}, \mathcal{R})$ be an AF. A* labelling *for $(\mathcal{A}, \mathcal{R})$ is a total function $\ell : \mathcal{A} \to \{\mathtt{in}, \mathtt{out}, \mathtt{und}\}$.*
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 [BCG11])**.** *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.*

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

- 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 [CG09]. 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.*

In the 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 [BCGG11]). In recent works (see for instance [CFFLS18]), 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.

The input consists of a dependence graph (nodes are arguments and edges are dependence links) together with an acceptance condition attached to each node. This condition is a propositional formula 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 whose nodes represent arguments and edges represent dependence links. The status of an argument depends on the status of its parents in the graph following the acceptance condition attached to the argument. This acceptance condition is a propositional formula. 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).[5]
>
> 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.

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,

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

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 fixpoint if it exists (it is always the case for three-valued operators).

Def. 8 (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.*

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. 9 (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. 10 (Preordering $\leq_t$ on two-valued interpretations)**.** *Let $(S, \{\varphi_s | s \in S\})$ be an ADF. Let v_1 and v_2 be two-valued interpretations. $v_1 \leq_t v_2$ if and only if $\forall s \in S$, $v_1(s) = t \Rightarrow v_2(s) = t$.*

Def. 11 (Preordering $\leq_i$ on three-valued interpretations)**.** *Let $(S, \{\varphi_s | s \in S\})$ be an ADF. Let v_1 and v_2 be three-valued interpretations. $v_1 \leq_i v_2$ 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:[6]

Def. 12. *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 $v \leq_i \Gamma_D(v)$.*
- *v is* preferred *for D iff v is $\leq_i$-maximal admissible.*
- *v is* grounded *for D iff v is the $\leq_i$-least fixpoint 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.*

[6]The definition for the stable models is not given here but can be found in [BES+18] (a *model* being an interpretation v such that $\forall s \in S$, $v(s) = v(\varphi_s)$).

- *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.*[7]

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 [Pol17].

- *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).

  - 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*);
  - The acceptance notion can also be related to a specific set of arguments that are not the target of a link (neither attacked, nor supported) and that are considered as accepted.

  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).

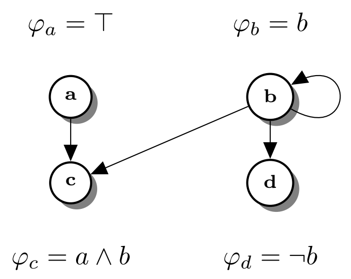
[7]Note that the grounded interpretation is unique.

- *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 [BCGG11] 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.

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. 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 Figure 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 fixpoint for the operator G_D. Note that, in this example, there exist 2 fixpoints (each fixpoint being by definition a two-valued model).

This example also illustrates the preordering $\leq_t$, see Figure 4 (given at the end of this paper). Note that the set of two-valued interpretations over $\leq_t$ 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,[8] *we do not represent the $\leq_i$ preordering and the corresponding complete meet-lattice. 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 fixpoint 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 fixpoint of G_D (so a two-valued model),*
- $v = \{a \mapsto t,\ b \mapsto f,\ c \mapsto f,\ d \mapsto u\}$.

Following Def. 12, v_0, v_1 and v_2 are complete (they are the only fixpoints of Γ_D). Moreover v_1 and v_2 are preferred and v_0 is grounded. It can be shown that v is admissible.[9]

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

Ex. 3.2.2. *Consider the ADF represented by:*

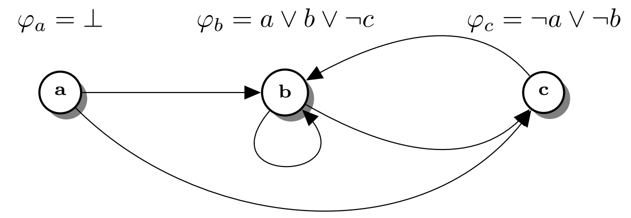

$\varphi_a = \bot$
$\varphi_b = a \vee b \vee \neg c$
$\varphi_c = \neg a \vee \neg b$
a
b
c

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).

[8] $3^4 = 81$ three-valued interpretations.

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

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':

- *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 fixpoints of this example.

In this example, the preordering $\leq_i$ is given by Figure 6 (given at the end of this paper). Note that the set of three-valued interpretations over $\leq_i$ 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).

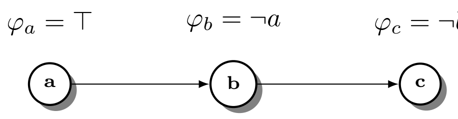
Here are some interesting three-valued interpretations:

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

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

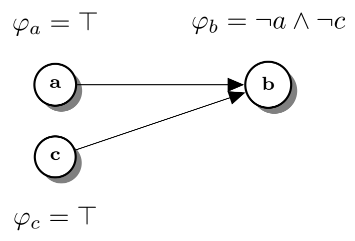
The following examples illustrate the use of ADF as a modelling tool.

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

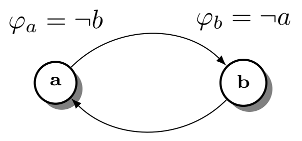


Applying the G_D or the Γ_D operators gives only one fixpoint: $v = \{a \mapsto t, b \mapsto f, c \mapsto t\}$. That corresponds to the complete, preferred and grounded extension.

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

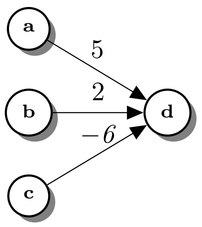


Ex. 3.2.5. *An even-length cycle of attacks can be translated into the following ADF:*



Applying the Γ_D operator gives three fixpoints: $v_0 = \{a \mapsto u, b \mapsto u\}$, $v_1 = \{a \mapsto t, b \mapsto f\}$ 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:

$$\begin{tikzcd}
\varphi_a = \top & \mathbf{a} \arrow[dr] & & \\
\varphi_b = \bot & \mathbf{b} \arrow[r] & \mathbf{d} & \varphi_d = (\neg c \wedge (a \vee b)) \vee (a \wedge b) \\
\varphi_c = \top & \mathbf{c} \arrow[ur] & &
\end{tikzcd}$$

Ex. 2.1.2 (cont'd) *Consider a BAF with only one support from a to b.*

If we consider that the support is a deductive one then the meaning of this support 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".

If we consider that the support is a necessary one then the meaning of this support can be described by the following assertion (the target of a support impacts its source):

"a is accepted if b is accepted".

In the first case (the deductive meaning of the support), the BAF could be represented by an ADF with one positive link between a and b with $\varphi_b = a$ and $\varphi_a = \top$. However, a positive link from a to b with $\varphi_b = a$ expresses a sufficient condition (the "if part" means "if a is accepted then b is accepted") and a necessary condition (the "only if" part means "a is accepted if b is accepted"). The sufficient condition encodes exactly the deductive meaning of the support but the necessary condition gives an additional constraint and we can easily see that this additional constraint corresponds to the necessary meaning of the support.

Consider now the necessary meaning of the support, the BAF could be represented by an ADF with one positive link from b to a with $\varphi_a = b$ and $\varphi_b = \top$.[10] *Once again, a positive link from b to a with $\varphi_a = b$ expresses a sufficient condition (the "if part" means "if b is accepted then a is accepted") and a necessary condition (the "only if" part means "b is accepted if a is accepted"). The sufficient condition encodes exactly the necessary support but the necessary condition gives an additional constraint and we can easily see that this additional constraint corresponds to the deductive meaning of the support.*

So, in both cases, 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"). However, the difference between

[10]This representation is obtained using the duality between deductive and necessary supports: a deductive support from a to b is exactly a necessary support from b to a and vice-versa.

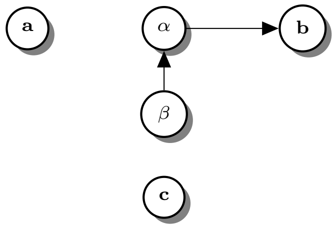
the two ADFs is that the formula is attached either to b (deductive support), or to a (necessary support).
Nevertheless, if there is no other interaction from, or to these two arguments, the acceptance conditions do not allow to distinguish between the two meanings of the supports.
A synthesis of the above remarks is given in the following table. Let us consider a support from a to b, denoted by the pair (a, b):

"if part" of $\varphi_b = a$	*"only if part" of $\varphi_b = a$*	*"if part" of $\varphi_a = b$*	*"only if part" of $\varphi_a = b$*
deductive meaning of (a, b)	*necessary meaning of (a, b)*	*necessary meaning of (a, b)*	*deductive meaning of (a, b)*

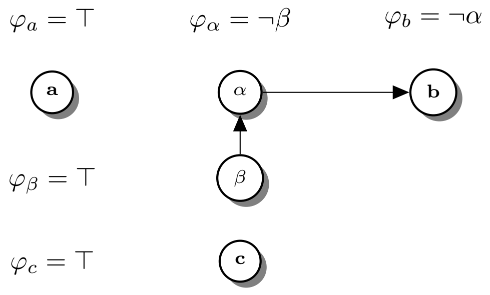
In conclusion, as the acceptance condition in an ADF reads as a necessary and sufficient condition, it seems difficult to capture a support which is not both necessary and deductive.

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:



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



3.3 Some implementations

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

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).[11] 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 9.5) 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. 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 (AF, BAF, ...).

The idea to attach an acceptance condition to an AF has also been used in [CMDM06]. 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:

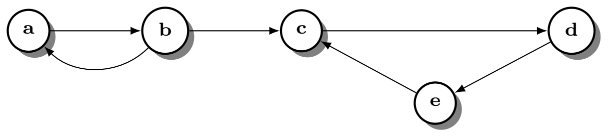
- the graph used in [CMDM06] is a standard argumentation graph (links encode attacks) and not a dependence graph,

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

- there is a unique acceptance condition in [CMDM06] and not one attached to each argument,
- [CMDM06] defines 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 (nodes of the argumentation graph are injectively mapped to some items, e.g., indexes, and edges can be injectively mapped to special propositional atoms doomed to be true). 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 [DHP14], $r_{a,b}$ in [DGLW15], ...[12]).

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 [CFLS17, CLS18a, CLS18b]. The definitions presented here are those given in the more recent paper [CLS18b].

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 [CLS18b] consists of the following axioms:

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

$$\Big(Acc(a) \to Nacc(b)\Big)$$

For every argument b in the input AF

$$\Big(Nacc(b) \to \neg Acc(b)\Big)$$

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

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

In the case of RAFs, the authors use a two-sorted logic with equality. There are `arg` a sort for arguments and `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 `att` to capture source and target of the attack. The target can be either of sort `arg` or of sort `att` and the source can only be of sort `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

$$\begin{array}{lll} \forall e : \texttt{att}, & \Big(\big(Val(e) \land Acc(s(e))\big) \to \neg Val(t(e))\Big) & \text{if } t(e) \text{ is of sort } \texttt{att} \\ \forall e : \texttt{att}, & \Big(\big(Val(e) \land Acc(s(e))\big) \to Nacc(t(e))\Big) & \text{if } t(e) \text{ is of sort } \texttt{arg} \end{array}$$

and the coherence axiom as above

$$\forall a : \texttt{arg}, \quad Nacc(a) \to \neg Acc(a)$$

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:

$$\begin{array}{rll} & s(e) = x \land t(e) = y & \text{for every edge from } x \text{ to } y \\ \forall a : \texttt{arg}, & a = a_1 \lor \cdots \lor a = a_n & \\ \forall e : \texttt{att}, & e = e_1 \lor \cdots \lor e = e_m & \\ & \neg(a_i = a_j) \land \neg(e_i = e_j) & i \neq j \end{array}$$

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.

4.3 The approach by Gabbay and Gabbay

[GG15] generates for every AF a theory with the axiom

$$Na \to \neg a$$

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:

$$a \to Nb$$

for all a and b such that $a\mathcal{R}b$.

The theory generated by [GG15] 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[13]
3. $\{z \to Ny | (z\mathcal{R}y)\}$
4. $\{(\bigwedge_{z\in\mathcal{R}^-(y)} \neg z \wedge \bigvee_{z\in\mathcal{R}^-(y)} \neg Nz) \to \neg y \wedge \neg Ny\}$ for each argument y[14]

The complete extensions correspond to the models of the above theory.

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

$$a \vee Na$$

for all a.

The corresponding stability axiom given in [CLS18b] is

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

Both [CFLS17] and [GG15] prove that the models of the respective theory coincide with the extensions of the AF (the list of semantics dealt with in [CFLS17] is longer).

4.4 Some implementations

The approach of [CFLS17, CLS18a, CLS18b] has been implemented in Grafix, a Java software for representing graphically argumentation frameworks with higher-order interactions (see [CLS]).

[13]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.*

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

5 QBF-formalization of AF

The aim of both approaches [AC12, DWW14] 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 [DWW15]) together with an extension-based semantics. The output is a quantified Boolean formula (QBF)[15] 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 doomed to be true), with quantification serving to express minimality/maximality wrt 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 assigned by the labellings (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 [DWW14].

5.1 Main definitions

We first describe the approach due to Arieli and Caminada [AC12, AC13].

Let us recall that a labelling ℓ for an AF $(\mathcal{A}, \mathcal{R})$, is a total function $\ell : \mathcal{A} \to \{\mathtt{in}, \mathtt{out}, \mathtt{und}\}$. 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
- 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 set of all complete labellings for $AF = (\mathcal{A}, \mathcal{R})$ is denoted by $Comp(AF)$.

[15]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: $\forall x \exists y \exists z((x \vee z) \wedge y)$.

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

Def. 13. *For an unsigned atom a and unsigned formulae ϕ, ψ, define the following signed formulae:*

$$\begin{array}{ll}
\tau^+(a) = a^{\oplus} & \tau^-(a) = a^{\ominus} \\
\tau^+(\neg\phi) = \tau^-(\phi) & \tau^-(\neg\phi) = \tau^+(\phi) \\
\tau^+(\phi \to \psi) = \neg\tau^-(\phi) \vee \tau^+(\psi) & \tau^-(\phi \to \psi) = \tau^+(\phi) \wedge \tau^-(\psi)
\end{array}$$

The encoding then expands to

Def. 14. *For an unsigned formula ϕ:*

$$\begin{array}{rcl}
\texttt{val}(\phi, t) & = & \tau^+(\phi) \wedge \neg\tau^-(\phi) \\
\texttt{val}(\phi, f) & = & \neg\tau^+(\phi) \wedge \tau^-(\phi) \\
\texttt{val}(\phi, u) & = & \neg\tau^+(\phi) \wedge \neg\tau^-(\phi)
\end{array}$$

The main definition is:[16]

Def. 15. *Let $AF = (\mathcal{A}, \mathcal{R})$ and $x \in \mathcal{A}$. Define*

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

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:

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

Also, stable extensions can be captured:

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

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:

$$\forall x_1^{\oplus} x_1^{\ominus} \cdots x_n^{\oplus} x_n^{\ominus}$$

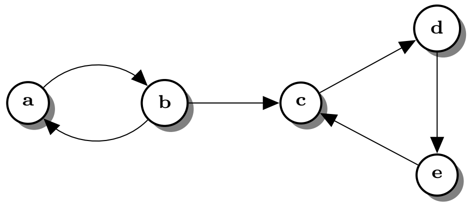
$$\left[CMP(AF)[x_1, \ldots, x_n] \to \left(\begin{array}{l} \left(\bigwedge_{a_i \in \mathcal{A}} \left(\texttt{val}(x_i,t) \to \texttt{val}(a_i,t)\right)\right) \\ \to \left(\bigwedge_{a_i \in \mathcal{A}} \left(\texttt{val}(a_i,t) \to \texttt{val}(x_i,t)\right)\right) \end{array} \right) \right]$$

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

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 $\bigwedge_{a_i \in \mathcal{A}} \big(\texttt{val}(x_i, t) \to \texttt{val}(a_i, t)\big)$ 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:[17]

$$\begin{aligned}
&\texttt{val}(a,t) \to \texttt{val}(b,f)\\
&\texttt{val}(b,t) \to \texttt{val}(a,f)\\
&\texttt{val}(c,t) \to \big(\texttt{val}(b,f) \wedge \texttt{val}(e,f)\big)\\
&\texttt{val}(d,t) \to \texttt{val}(c,f)\\
&\texttt{val}(e,t) \to \texttt{val}(d,f)
\end{aligned}$$

$$\begin{aligned}
&\texttt{val}(a,f) \to \texttt{val}(b,t)\\
&\texttt{val}(b,f) \to \texttt{val}(a,t)\\
&\texttt{val}(c,f) \to \big(\texttt{val}(b,t) \vee \texttt{val}(e,t)\big)\\
&\texttt{val}(d,f) \to \texttt{val}(c,t)\\
&\texttt{val}(e,f) \to \texttt{val}(d,t)
\end{aligned}$$

$$\begin{aligned}
&\texttt{val}(a,u) \to \Big(\neg\texttt{val}(b,f) \wedge \neg\texttt{val}(b,t)\Big)\\
&\texttt{val}(b,u) \to \Big(\neg\texttt{val}(a,f) \wedge \neg\texttt{val}(a,t)\Big)\\
&\texttt{val}(c,u) \to \Bigg(\neg\Big(\texttt{val}(b,f) \wedge \texttt{val}(e,f)\Big) \wedge \neg\Big(\texttt{val}(b,t) \vee \texttt{val}(e,t)\Big)\Bigg)\\
&\texttt{val}(d,u) \to \Big(\neg\texttt{val}(c,f) \wedge \neg\texttt{val}(c,t)\Big)\\
&\texttt{val}(e,u) \to \Big(\neg\texttt{val}(d,f) \wedge \neg\texttt{val}(d,t)\Big)
\end{aligned}$$

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

In the signed language,

$$\begin{array}{l}
(a^\oplus \wedge \neg a^\ominus) \to (b^\ominus \wedge \neg b^\oplus)\\
(b^\oplus \wedge \neg b^\ominus) \to (a^\ominus \wedge \neg a^\oplus)\\
(c^\oplus \wedge \neg c^\ominus) \to \Big((b^\ominus \wedge \neg b^\oplus) \wedge (e^\ominus \wedge \neg e^\oplus)\Big)\\
(d^\oplus \wedge \neg d^\ominus) \to (c^\ominus \wedge \neg c^\oplus)\\
(e^\oplus \wedge \neg e^\ominus) \to (d^\ominus \wedge \neg d^\oplus)
\end{array}$$

$$\begin{array}{l}
(a^\ominus \wedge \neg a^\oplus) \to (b^\oplus \wedge \neg b^\ominus)\\
(b^\ominus \wedge \neg b^\oplus) \to (a^\oplus \wedge \neg a^\ominus)\\
(c^\ominus \wedge \neg c^\oplus) \to \Big((b^\oplus \wedge \neg b^\ominus) \vee (e^\oplus \wedge \neg e^\ominus)\Big)\\
(d^\ominus \wedge \neg d^\oplus) \to (c^\oplus \wedge \neg c^\ominus)\\
(e^\ominus \wedge \neg e^\oplus) \to (d^\oplus \wedge \neg d^\ominus)
\end{array}$$

$$\begin{array}{l}
(\neg a^\oplus \wedge \neg a^\ominus) \to \Big(\neg(b^\ominus \wedge \neg b^\oplus) \wedge \neg(b^\oplus \wedge \neg b^\ominus)\Big)\\
(\neg b^\oplus \wedge \neg b^\ominus) \to \Big(\neg(a^\ominus \wedge \neg a^\oplus) \wedge \neg(a^\oplus \wedge \neg a^\ominus)\Big)\\
((\neg c^\oplus \wedge \neg c^\ominus) \to \Big[\neg\Big((b^\ominus \wedge \neg b^\oplus) \wedge (e^\ominus \wedge \neg e^\oplus)\Big) \wedge \neg\Big((b^\oplus \wedge \neg b^\ominus) \vee (e^\oplus \wedge \neg e^\ominus)\Big)\Big]\\
\neg d^\oplus \wedge \neg d^\ominus) \to \Big(\neg(c^\ominus \wedge \neg c^\oplus) \wedge \neg(c^\oplus \wedge \neg c^\ominus)\Big)\\
(\neg e^\oplus \wedge \neg e^\ominus) \to \Big(\neg(d^\ominus \wedge \neg d^\oplus) \wedge \neg(d^\oplus \wedge \neg d^\ominus)\Big)
\end{array}$$

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

5.3 A QBF approach for ADF

The QBF-based formalization due to Diller, Wallner and Woltran [DWW14, DWW15] concerns ADF.

This approach also uses a signed language, with the same atoms as given in Def. 13, 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 $a \in \mathcal{A}$:

$$\neg(a^\oplus \wedge a^\ominus)$$

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:

$$\forall x_1^\oplus x_1^\ominus \cdots x_n^\oplus x_n^\ominus$$

$$\left[\bigwedge_{a\in\mathcal{A}} \Big(\Big(a^\oplus \to a\Big) \wedge \Big(a^\ominus \to \neg a\Big)\Big) \to \bigwedge_{a\in\mathcal{A}} \Big(\Big(a^\oplus \to \varphi_a\Big) \wedge \Big(a^\ominus \to \neg\varphi_a\Big)\Big)\right]$$

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)

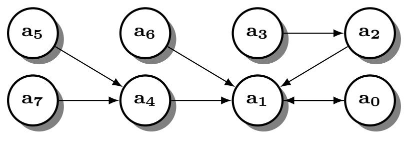
The aim of this approach is the definition of a first-order logical theory capable of describing an AF and its standard semantics. In the basic language YALLA, an AF is described by specific axioms of the theory and formulae are interpreted by argumentation graphs. A variant of the basic language called 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, 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

[DBCL16] 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 formulas of the base. In the following example, it is assumed that $\mathcal{A}_{\mathbf{U}}$ and $\mathcal{R}_{\mathbf{U}}$ are explicitly provided:

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 [DBCL16], 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 [Dun95] in the sense that arguments and interactions must be built according to the universe.

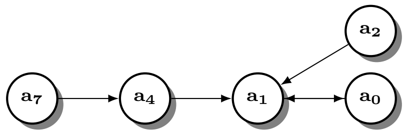
Def. 16. *An AF on* $(\mathcal{A}_\mathbf{U}, \mathcal{R}_\mathbf{U})$ *is a pair* $(\mathcal{A}, \mathcal{R})$ *where*

- $\mathcal{A} \subseteq \mathcal{A}_\mathbf{U}$ *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})*:*



[DBCL16] 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 $\mathcal{A}_\mathbf{U} = \{a_1, a_2, \cdots, a_k\}$ with k being the cardinal of $\mathcal{A}_\mathbf{U}$. The signature of the associated language YALLA$_\mathbf{U}$ is defined as follows:[18]

Def. 17 (Signature). $\Sigma_\mathbf{U} = (V_{const}, V_f, V_P)$ *where the set of constants* $V_{const} = \{c_\perp, c_1, \ldots, c_p\}$ *with* $p = 2^k - 1$*, the set of functions* $V_f = \{union^2\}$ *and the set of predicates* $V_P = \{on^1, \rhd^2, \subseteq^2\}$.

As the logical theory has been built for describing AFs on a given universe, terms and formulas of the language YALLA$_\mathbf{U}$ will be interpreted on AFs built on this universe. Formally, the semantics of YALLA$_\mathbf{U}$ is defined thanks to a structure over $\Sigma_\mathbf{U}$, on which terms and formulas 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.

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

Def. 18 (**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 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 YALLA$_{\mathbf{U}}$.

Def. 19 (**Formula describing an AF**). *Let the function* $\Phi_{\mathbf{U}}$ *be defined as follows:*

$$\begin{array}{llll} \Phi_{\mathbf{U}}: & \Gamma_{\mathbf{U}} & \to & \mathit{YALLA}_{\mathbf{U}} \\ & (\mathcal{A}, \mathcal{R}) & \mapsto & on(\mathcal{A}) \quad \wedge \\ & & & \bigwedge_{x \in \mathcal{A}_{\mathbf{U}} \setminus \mathcal{A}} \neg on(\{x\}) \wedge \\ & & & \bigwedge_{(x,y) \in \mathcal{R}} (\{x\} \rhd \{y\}) \quad \wedge \\ & & & \bigwedge_{(x,y) \in \mathcal{R}_{\mathbf{U}} \setminus \mathcal{R}} \neg(\{x\} \rhd \{y\}) \end{array}$$

$\Phi_{\mathbf{U}}(\mathcal{A}, \mathcal{R})$ *is called the* characteristic formula of $(\mathcal{A}, \mathcal{R})$.[19]

Note that $(\mathcal{A}, \mathcal{R})$ determines the unique structure (Def. 18) 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 YALLA$_{\mathbf{U}}$,

$$\begin{array}{rcl} t_1 = t_2 & \overset{\text{def}}{\equiv} & (t_1 \subseteq t_2) \wedge (t_2 \subseteq t_1). \\ t_1 \neq t_2 & \overset{\text{def}}{\equiv} & \neg(t_1 = t_2). \\ singl(t_1) & \overset{\text{def}}{\equiv} & (t_1 \neq c_\perp) \wedge \forall t_2\ (((t_2 \neq c_\perp) \wedge (t_2 \subseteq t_1)) \implies (t_1 \subseteq t_2)). \end{array}$$

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 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* $\mathit{YALLA}_{\mathbf{U}}$.

- 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* $t_1 \ggg t_2$.
- t *is* admissible *in* $(\mathcal{A}, \mathcal{R})$ *if and only if* $(\mathcal{A}, \mathcal{R}) \models (F(t) \wedge (t \ggg t))$. *The latter formula is denoted by* $A(t)$.

[19]When t denotes a term of 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* 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 $G(t)$.*

- *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 YALLA$_{\mathbf{U}}$ formula can be viewed as a propositional formula and the satisfiability problem of a YALLA$_{\mathbf{U}}$ base is a NP-complete problem.

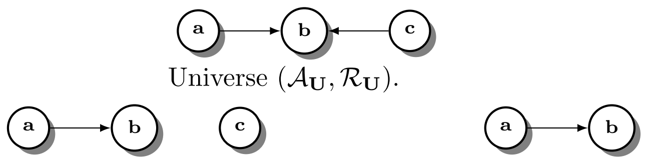
In [DBCL16] YALLA$_{\mathbf{U}}$ is used for expressing update in argumentation dynamics. For instance, in the case of a debate, classical update amounts to consider a formula φ in YALLA$_{\mathbf{U}}$ representing a current state of knowledge about exchanged arguments (i.e., it may encompass several 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 YALLA$_{\mathbf{U}}$ $\times$ YALLA$_{\mathbf{U}}$ to YALLA$_{\mathbf{U}}$ which associates with any formula φ and any formula α a formula, denoted by $\varphi \diamondsuit_{\mathcal{T}} \alpha$ satisfying $\mathcal{T}$. In [DBCL16], 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 [CDLS10, BCDLS13].

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 $\mathcal{A}_2$.



Universe $(\mathcal{A}_{\mathbf{U}}, \mathcal{R}_{\mathbf{U}})$.

Argumentation graph $(\mathcal{A}_1, \mathcal{R}_1)$. Argumentation graph $(\mathcal{A}_2, \mathcal{R}_2)$.

Figure 1: An example of argumentation graphs built on a universe

Moreover, following Definition 19, we have:[20]

$$\Phi_{\mathbf{U}}(\mathcal{A}_1, \mathcal{R}_1) = on(\{a,b,c\}) \land (\{a\} \rhd \{b\}) \land \neg(\{c\} \rhd \{b\})$$
$$\Phi_{\mathbf{U}}(\mathcal{A}_2, \mathcal{R}_2) = on(\{a,b\}) \land \neg(on(\{c\})) \land (\{a\} \rhd \{b\}) \land \neg(\{c\} \rhd \{b\})$$

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 $(\mathcal{A}_1, \mathcal{R}_1)$:

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

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 Figure 2.*

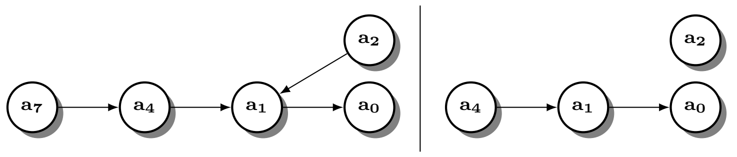


Figure 2: Two possible cases for the argumentation graph of Agent Ag_b

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

$$\begin{array}{rcrcrcrc}
\varphi & = & & & on(\{a_0,a_1,a_2,a_4\}) & \land & & \\
& & \neg(on(\{a_3\})) & \land & \neg(on(\{a_5\})) & \land & \neg(on(\{a_6\})) & \land \\
& & (\{a_4\} \rhd \{a_1\}) & \land & (\{a_1\} \rhd \{a_0\}) & \land & & \\
& & \neg(\{a_0\} \rhd \{a_1\}) & \land & \neg(\{a_3\} \rhd \{a_2\}) & \land & \neg(\{a_5\} \rhd \{a_4\}) & \land \\
& & \neg(\{a_6\} \rhd \{a_1\}) & \land & & & & \\
& \left(\begin{array}{r} \\ \\ \\ \end{array}\right. & (on(\{a_7\}) & \land & (\{a_7\} \rhd \{a_4\}) & \land & (\{a_2\} \rhd \{a_1\})) & \\
& & & & & \lor & & \\
& & (\neg(on(\{a_7\})) & \land & \neg(\{a_7\} \rhd \{a_4\}) & \land & \neg(\{a_2\} \rhd \{a_1\})) & \left.\begin{array}{r} \\ \\ \\ \end{array}\right)
\end{array}$$

[20]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 $\mathbf{U}$.

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 [BKRv13, DHP14, CMKMM14a, CMKMM14b, DHL$^+$15].

6.2 The approach by Villata et al.

[VBG$^+$12] proposes 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.

Contrastedly, YALLA is a first-order language with more expressivity, able to encode sets of arguments, set-theoretic properties (inclusion, union) and attacks between sets of arguments. Thus, YALLA makes it possible 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.

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. Both approaches are theoretical as well as practical: implementations exist.

7.1 The approach by Besnard and Doutre

The original proposal is [BD04], extended in [BDH14]. Independently, an equivalent proposal is [WD12].

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

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

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:

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

As to the case of complete extensions,

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

7.1.2 Some implementations

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

In [BDHL16, BDH14] and thanks to the project SESAME (see [BBD$^+$]), 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 [WWW13], the encoding proposed by Besnard and Doutre in [BD04] 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 [LLM15], the encoding proposed by Besnard and Doutre in [BD04] 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 [LNJ18], the encoding proposed by Besnard and Doutre in [BD04] 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 [DJWW12], 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 $(\mathcal{A}, \mathcal{R})$,

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

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.

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 [DJWW]) 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[21] 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 [CG09], 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 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.

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 $AF = (\mathcal{A}, \mathcal{R})$:

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

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 [CDGV14], 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*:

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

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

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:

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

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 [CDGV14] and [CG09] can be matched, owing to the equivalence between the alternative definitions of complete labellings provided in [CG09] (see Section 2.2).

8.3 Some implementations

At least three systems have been developped using the approach proposed in [CDGV14].

In [CDGV14], an algorithm using a SAT solver is proposed for computing the preferred semantics. This algorithm is called PrefSAT.

In [CGV14], 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 [BBP15], 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 [CDGV14].

9 Argumentation Frameworks and Logic Programming

The purpose of the works reviewed in this section is to associate a logic program with an AF in such a way that logic programming semantics (stable model semantics, P-stable model semantics, regular model semantics, ...), applied to the logic program capture semantics (stable, complete, preferred, ...) of the AF.
Some of the considered works encode an AF into a logic program. Other works consider the other way and encode a normal logic program into an AF. There also exist some generalizations to enriched argumentation frameworks (BAF, RAF, ...).

9.1 Introduction

The close connection between AF semantics and Logic Programming (LP) semantics goes back to Dung's work [Dun95]. 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 [WCG09, CSAD15].

Roughly speaking, the purpose of these works is to provide an argumentative semantics for logic programs.

On the other side, Dung 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 also been extended to relate other AF semantics to LP semantics [EGW10, CNO09, ONS13, CSAD15, ON17, SR17].

In that case, the purpose is rather to apply computational techniques of Logic Programming to argumentation.

The different works reviewed in this section can be distinguished according to the following features:

- The transformation is from AF to LP, or from LP to AF.
- AF semantics can be encoded under the form of extension-based semantics as in [Dun95, ONS13], or under the form of labelling-based semantics (as for instance in [CSAD15, SR17]).
- Different semantics of AF correspond to different semantics of a same logic program as in [Dun95, CSAD15] or different semantics of AF are all characterized by the single 2-valued stable model semantics of different transformed programs as in [SR17].

Moreover, some of these works have considered the issue of using ASP-solvers to compute the extensions of AFs under different semantics [EGW10, Gag10].

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 and vice-versa, all of them offering different characterizations of argumentation semantics in terms of LP semantics. A summary of these characterizations can be found in [ON17] with all the associated references.

In the following, we first consider some works relating an AF and Logic Programs, divided in two types:

1. Encoding an AF into a logic program.

2. Encoding a normal logic program into an AF.

Then we consider generalizations to enriched argumentation frameworks (BAF, RAF).

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.

9.2 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 [Dun95, CNO09, ONS13]) or in an implicit way (as for instance in [CSAD15]).

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 [Dun95]. 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 [CNO09], [ONS13], [ON17] 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:

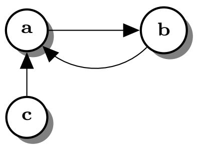
- The logic program includes weak negation (i.e. negation as failure, with the symbol *not*).
- The encoding captures both conflict-freeness and admissibility principles.
- The predicate *def* is used. $def(x)$ means "x is defeated" or equivalently "x cannot belong to an admissible set".
- The logic 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) *The associated logic program is:*
$def(b) \leftarrow not\ def(a)$
$def(c) \leftarrow not\ def(b)$
$def(b) \leftarrow \top$
$def(c) \leftarrow def(a)$
$acc(x) \leftarrow not\ def(x)$

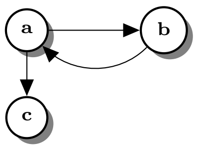
Ex. 3.2.4 (cont'd) *The associated logic program is:*
$def(b) \leftarrow not\ def(a)$
$def(b) \leftarrow not\ def(c)$
$def(b) \leftarrow \top$
$acc(x) \leftarrow not\ def(x)$

Ex. 9.2.1. *Consider the following AF:*



The associated logic program is:
$def(b) \leftarrow not\ def(a)$
$def(a) \leftarrow not\ def(b)$
$def(a) \leftarrow not\ def(c)$
$def(b) \leftarrow def(b), def(c)$
$def(a) \leftarrow def(a)$
$def(a) \leftarrow \top$
$acc(x) \leftarrow not\ def(x)$

Ex. 9.2.2. *Consider the following AF:*



The associated logic program is:
$def(b) \leftarrow not\ def(a)$
$def(a) \leftarrow not\ def(b)$
$def(c) \leftarrow not\ def(a)$
$def(b) \leftarrow def(b)$
$def(a) \leftarrow def(a)$
$def(c) \leftarrow def(b)$
$acc(x) \leftarrow not\ def(x)$

Ex. 3.4.1 (cont'd) *The associated logic program is:*
$def(a) \leftarrow not\ def(b)$
$def(b) \leftarrow not\ def(a)$
$def(c) \leftarrow not\ def(b)$
$def(c) \leftarrow not\ def(e)$
$def(d) \leftarrow not\ def(c)$
$def(e) \leftarrow not\ def(d)$
$def(a) \leftarrow def(a)$
$def(b) \leftarrow def(b)$
$def(c) \leftarrow def(a)$
$def(c) \leftarrow def(d)$
$def(d) \leftarrow def(b), def(e)$
$def(e) \leftarrow def(c)$

$acc(x) \leftarrow not\ def(x)$
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 [SR17] 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:

- $in(x) \leftarrow out(y_1), \ldots, out(y_n)$, 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:*

$$\begin{array}{l|l|l}
in(c) \leftarrow out(b); & in(b) \leftarrow out(a); & in(a) \leftarrow; \\
out(b) \leftarrow in(a); & out(c) \leftarrow in(b); & \\
\leftarrow in(c), not\ out(b); & \leftarrow in(b), not\ out(a); & \\
\leftarrow out(c), not\ in(b); & \leftarrow out(b), not\ in(a); & \leftarrow out(a);
\end{array}$$

$und(x) \leftarrow not\ in(x), not\ out(x)$ *(for* $x \in \{a, b, c\}$*).*
The unique stable model is $\{in(a), out(b), in(c)\}$.

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

$$\begin{array}{l|l|l}
in(c) \leftarrow; & in(b) \leftarrow out(a), out(c); & in(a) \leftarrow; \\
out(b) \leftarrow in(a); & out(b) \leftarrow in(c); & \\
\leftarrow in(b), not\ out(a); & \leftarrow in(b), not\ out(c); & \\
\leftarrow out(c), not\ in(b); & \leftarrow out(b), not\ in(a), not\ in(c); & \\
\leftarrow out(a); & \leftarrow out(c); &
\end{array}$$

$und(x) \leftarrow not\ in(x), not\ out(x)$ *(for* $x \in \{a, b, c\}$*).*
The unique stable model is $\{in(a), out(b), in(c)\}$.

Ex. 9.2.1 (cont'd) *The associated logic program under the grounded semantics is:*

$$\begin{array}{l|l|l}
in(a) \leftarrow out(b), out(c); & in(b) \leftarrow out(a); & in(c) \leftarrow; \\
out(a) \leftarrow in(b); & out(a) \leftarrow in(c); & \\
out(b) \leftarrow in(a); & \leftarrow in(a), not\ out(b); & \\
\leftarrow in(a), not\ out(c); & \leftarrow in(b), not\ out(a); & \\
\leftarrow out(a), not\ in(b), not\ in(c); & \leftarrow out(b), not\ in(a); & \leftarrow out(c);
\end{array}$$

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

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

Ex. 9.2.2 (cont'd) *The associated logic program under the grounded semantics is:*

$$\begin{array}{l|l|l}
in(a) \leftarrow out(b); & in(b) \leftarrow out(a); & in(c) \leftarrow out(a); \\
out(a) \leftarrow in(b); & out(b) \leftarrow in(a); & out(c) \leftarrow in(a); \\
\leftarrow in(a), not\ out(b); & \leftarrow in(b), not\ out(a); & \leftarrow in(c), not\ out(a); \\
\leftarrow out(a), not\ in(b); & \leftarrow out(b), not\ in(a); & \leftarrow out(c), not\ in(a);
\end{array}$$

$und(x) \leftarrow not\ in(x), not\ out(x)$ *(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 [CSAD15]. 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 $x \leftarrow not\ y_1, \dots, not\ y_n$, where $y_1, \dots, y_n$ 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:*

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

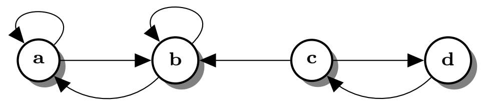
$a \leftarrow$
$c \leftarrow$

Ex. 9.2.1 (cont'd) *The associated logic program is:*
$b \leftarrow not\ a$
$a \leftarrow not\ b, not\ c$
$c \leftarrow$

Ex. 9.2.2 (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. 9.2.3. *Consider the following AF:*



The associated logic program is:
$a \leftarrow not\ a, not\ b$
$b \leftarrow not\ b, not\ a, not\ c$
$c \leftarrow not\ d$
$d \leftarrow not\ c$

9.3 From a logic program to an AF

The input is a logic program. The output consists of an AF together with the characterization of some of the standard argumentation semantics using the models of the original logic program.

Note that, in contrast with the other works reported in this survey, the resulting argumentation framework is not, strictly speaking, abstract.

The characteristic features of the approach described in [CSAD15] are:

- From a logic program P, an instantiated AF is produced where an argument is built from rules of P, and has a conclusion.

- From standard argumentation semantics, new semantics are defined for selecting argument conclusions in the instantiated AF.

- Correspondences between those new semantics and three-valued models of the original program are given.

Let us first detail the construction of the AF.
An argument built from the logic program P can be seen as a tree-like structure of rules, obtained by chaining a subset of rules of P. An argument involves a set of rules, a conclusion (the head of the last used rule), a set of "vulnerabilities" (the atoms appearing in the weak part of the rules involved in the argument) and a set of sub-arguments. Then an attack is defined from an argument A to an argument B if the conclusion of A belongs to the set of vulnerabilities of B.

Ex. 9.3.1. *Consider the following normal logic program P with rules r_1, r_2:*
$r1 : a \leftarrow b$*;* $\quad r2 : b \leftarrow a$*.*
The following arguments are constructed:
$A_1 : a \leftarrow (A_2)$*;* $\quad A_2 : b \leftarrow (A_1)$*.*
As there is no weak part in the rules, there is no attack between the arguments.

Ex. 9.3.2. *Consider the following normal logic program P with rules $r_1, \ldots r_6$:*

$r1 : b \leftarrow c, not\ a;$	$r2 : a \leftarrow not\ b;$	$r3 : p \leftarrow c, d, not\ p;$
$r4 : p \leftarrow not\ a;$	$r5 : c \leftarrow d;$	$r6 : d \leftarrow .$

The following arguments are constructed:

$A_1 : d \leftarrow;$	$A_2 : c \leftarrow (A_1);$	$A_3 : p \leftarrow (A_2), (A_1), not\ p;$
$A_4 : a \leftarrow not\ b;$	$A_5 : p \leftarrow not\ a;$	$A_6 : b \leftarrow (A_2), not\ a.$

Note that the set of vulnerabilities of A_1 (resp. A_2, A_3, A_4, A_5 , A_6) is $\varnothing$ (resp. $\varnothing$, $\{p\}$, $\{b\}$, $\{a\}$, $\{a\}$).
The attacks are as follows:
A_3 *attacks* A_3*;* A_6 *attacks* A_4*;* A_4 *attacks* A_5 *and* A_6*;* A_5 *attacks* A_3*.*

Then, labelling-based semantics are converted into conclusion labellings. The idea is for each conclusion to identify the "best" argument that yields it, that is the argument with the highest label. If there is no argument for a particular atom, this atom is labelled "out". Formally, a conclusion labelling is defined as follows :
Let P be a logic program, H be the set of atoms occurring in P and AF be the instantiated argumentation framework associated with P. Let $ArgL$ be an argument labelling of AF. The conclusion labelling associated with $ArgL$ is the function $ConcL$ from H to the set of labels $\{in, out, undec\}$ such that for each $c \in$ H, it holds that $ConcL(c) = max(\{ArgL(A)|c$ is the conclusion of $A\}$ $\cup\{out\})$ where $in > undec > out$.

Ex. 9.3.2 (cont'd) *There are three complete argument labellings of the AF. One if them is such that $in(ArgL) = \{A_1, A_2, A_5, A_6\}$, $out(ArgL) = \{A_3, A_4\}$, $undec(ArgL) = \{\}$.*
The atom p is the conclusion of arguments A_3 and A_5. So the label of p in the associated conclusion labelling is $max(in, out) = in$.

Similarly, we obtain that $in(ConcL) = \{c, d, b, p\}$, $out(ConcL) = \{a\}$ *and* $undec(ConcL) = \{\}$.

Note that if the original program P is simple (at most one rule whose head is a for each atom a of the program), and is such that each rule has only a weak part, then each argument constructed consists of exactly one rule.

Ex. 9.3.3. *Consider the following normal logic program P with rules* $r_1, \dots r_4$*:*

$r1 : a \leftarrow not\ a, not\ b;$	$r2 : c \leftarrow not\ d;$
$r3 : d \leftarrow not\ c;$	$r4 : b \leftarrow not\ b, not\ a, not\ c.$

As P is simple, each argument constructed consists of exactly one rule. The attacks are as follows:

A attacks A;	*B attacks B;*	*A attacks B;*
B attacks A;	*C attacks B and D;*	*D attacks C.*

Note that we obtain the framework pictured in Example 9.2.3.

Moreover, given an AF, let P denote the logic program obtained with the approach of [CSAD15] described in Section 9.2. Then consider the instantiated argumentation framework AF′ obtained from P with the approach of [CSAD15] described in Section 9.3. As P is a simple program whose rules have only a weak part, each argument of AF′ is reduced to one rule and the attack relation of AF′ is the same as the attack relation of the original framework AF. So it was proved in [CSAD15] that both argumentation frameworks AF and AF′ are isomorphic. This point is illustrated by both Example 9.2.3 and Example 9.3.3.

9.4 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.

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.

An AFN [Nou13] 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.

(1) 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.

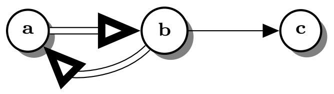
- For each set of arguments E that necessary supports an argument, an atom e is created.
- The body of the rule r_a contains all the weak atoms *not* b_i, with b_i attacks a, and all the strong atoms e_j, with E_j supports a.
- For each support E, there is a rule $e \leftarrow x$ for each $x \in E$.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

Ex. 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:*
$b \leftarrow not\ a, not\ c$
$a \leftarrow$
$c \leftarrow$

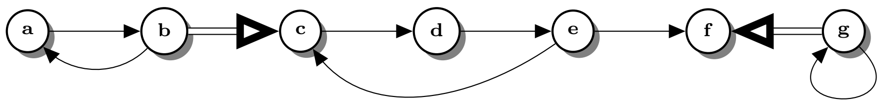
Ex. 3.4.1 (cont'd) *The associated logic program is:*
$b \leftarrow not\ a$
$a \leftarrow not\ b$
$c \leftarrow not\ b, not\ e$
$d \leftarrow not\ c$
$e \leftarrow not\ d$

Ex. 9.4.1. *Consider the following AFN:*



The associated logic program is:
$b \leftarrow e_1$
$a \leftarrow e_2$
$c \leftarrow not\ b$
$e_1 \leftarrow a$
$e_2 \leftarrow b$

Ex. 9.4.2. *Consider the following AFN:*

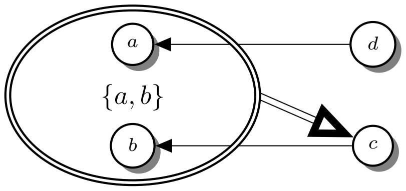


The associated logic program is:

$a \leftarrow not\ b$
$b \leftarrow not\ a$
$c \leftarrow e_1, not\ e$
$d \leftarrow not\ c$
$e \leftarrow not\ d$
$f \leftarrow e_2, not\ e$
$g \leftarrow not\ g$
$e_1 \leftarrow b$
$e_2 \leftarrow g$

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.*

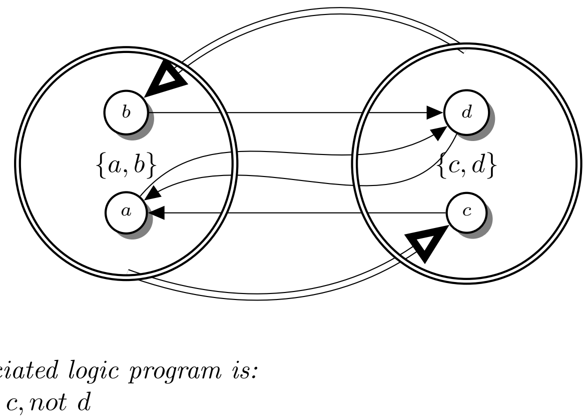
Ex. 9.4.3. *Consider the following AFN with a collective support:*



The associated logic program is:

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

Ex. 9.4.4. *Consider the following AFN with collective supports:*



The associated logic program is:

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

$b \leftarrow e_2$
$c \leftarrow e_1$
$d \leftarrow not\ a, not\ b$
$e_1 \leftarrow a$
$e_1 \leftarrow b$
$e_2 \leftarrow c$
$e_2 \leftarrow d$

(2) A logic program is encoded into an AFN as follows:

- An argument is associated with each rule of the program.
- The supports are created: if E is a set of rules with the same head h, and h is a strong atom of a rule r, then a necessary support from E to r is created.
- The attacks are created: if the head of r_1 appears in a weak atom of r_2, then an attack from r_1 to r_2 is created.
- There are correspondences between the three-valued of the program and the labelling-based semantics of the AFN (except for semi-stable semantics).

Ex. 9.3.1 (cont'd) *The logic program is encoded by the following AFN. An argument r_i is associated with each rule. There is no attack. The supports are as follows:*
$\{r_1\}$ *supports* r_2*;* $\{r_2\}$ *supports* r_1*.*

Ex. 9.3.2 (cont'd) *The logic program is encoded by the following AFN. An argument r_i is associated with each rule.*
The attacks are as follows:

r_3 *attacks* r_3*;*	r_1 *attacks* r_2*;*	r_2 *attacks* r_1*;*
r_2 *attacks* r_4*;*	r_4 *attacks* r_3*.*	

The supports are as follows:
$\{r_5\}$ *supports* r_1 *and* r_3*;* $\{r_6\}$ *supports* r_5*.*

Ex. 9.4.5. *Consider the following normal logic program P with rules $r_1, \ldots r_3$:*
$r1 : p \leftarrow q;$ $r2 : q \leftarrow p;$ $r3 : s \leftarrow not\ q.$
An argument r_i is associated with each rule. There is one attack:
r_2 *attacks* r_3*,*
and two supports:
$\{r_1\}$ *supports* r_2*,* $\{r_2\}$ *supports* r_1*.*
Note that we obtain the framework pictured in Example 9.4.1.

Ex. 9.4.6. *Consider the following normal logic program P with rules $r_1, \ldots r_7$:*
$r1 : p \leftarrow not\ q;$ $r2 : q \leftarrow not\ p;$ $r3 : s \leftarrow q, not\ u;$
$r4 : t \leftarrow not\ s;$ $r5 : u \leftarrow not\ t;$ $r6 : v \leftarrow w, not\ u;$

$r7 : w \leftarrow not\ w$.
An argument r_i is associated with each rule. It can be checked that the framework pictured in Example 9.4.2 is obtained.

Ex. 9.4.7. *Consider the following normal logic program P with rules $r_1, \dots r_4$:*
$r1 : p \leftarrow q$; $r2 : q \leftarrow not\ p$; $r3 : q \leftarrow not\ s$; $r4 : s \leftarrow$.
An argument r_i is associated with each rule. There are two attacks:
r_1 *attacks* r_2, r_4 *attacks* r_3
and one support:
$\{r2, r3\}$ *supports* r_1.
Note that we obtain the framework pictured in Example 9.4.3.

Ex. 9.4.8. *Consider the following normal logic program P with rules $r_1, \dots r_4$:*
$r1 : p \leftarrow not\ q$; $r2 : p \leftarrow q$; $r3 : q \leftarrow not\ p$; $r4 : q \leftarrow p$.
An argument r_i is associated with each rule. It can be checked that the framework pictured in Example 9.4.4 is obtained.

Note that if the original program P is simple (at most one rule whose head is a for each atom a of the program), and is such that each rule has only weak atoms, the encoding proposed by [CSAD15] is recovered.

The above remark is illustrated on the following example.

Ex. 9.3.3 (cont'd) *The logic program is encoded by the following AFN. An argument r_i is associated with each rule. The attacks are as follows:*

r_1 *attacks* r_1 *and* r_4 ;	r_2 *attacks* r_3 *and* r_4 ;
r_3 *attacks* r_2;	r_4 *attacks* r_1 *and* r_4.

There are no supports.

In contrast, when the logic program is not simple or when some rules have a strong part, the encoding by an AFN, using necessary supports, avoids the construction of complex arguments. The transformation from a program to an AFN is performed at the abstract level.

REBAF and logic programs Connections between a Recursive Evidence-Based Argumentation Framework (REBAF) and a logic program can also be considered (private communication by J. Fandinno).

A REBAF [CFFLS18] 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. Let us recall that evidential support is based on the intuition that every argument must be supported by some chain of supports from some special arguments called *prima-facie*.

A logic program can be encoded into a REBAF as follows:

Let r be the rule $h \leftarrow b_1, \dots, b_n, not\ c_1, \dots, not\ c_m$. The following REBAF is created:

- The arguments $h, b_1, \dots, b_n, c_1, \dots, c_m$ are created.

- Let $E = \{b_1, \dots b_n\}$. An evidential support named α_r is created from E to h. The source of α_r is the set of arguments E and the target of α_r is the argument h.
- For each $i = 1, \dots, m$, an attack named β_i is created, such that the source of β_i is $\{c_i\}$ and the target of β_i is α_r.
- All the attacks and supports are prima-facie. In contrast, no argument is prima-facie.

Preliminary work suggests that a correspondence between the stable models of the original program and the stable structures of the obtained REBAF could be obtained.

Up to now, the other direction (from a REBAF to a logic program) has not been considered.

The following particular case is worth of interest : Consider a simple logical program such that each rule has only a weak part.
In that case, each argument is considered as prima-facie (as a support would be created with an empty source).
Moreover, given the rule $r = h \leftarrow \textit{not}\ c_1, \dots, \textit{not}\ c_m$, each attack β_i should be created with the target h. Note that in that particular case, with the encoding proposed in [CSAD15], each argument which is created consists of exactly one rule. In contrast, in the encoding proposed above, arguments are exactly atoms of the program.

Let us compare both proposals on the following example:

Ex. 9.3.3 (cont'd) *Let us first consider the normal logic program P reduced to the rule*

$r1 : a \leftarrow \textit{not}\ a, \textit{not}\ b.$

With the encoding of [CSAD15], only one argument A_1 is created, consisting of the unique rule $r1$. There is only one attack:

$A1$ *attacks* $A1$.

In contrast, with the REBAF approach, two prima-facie arguments a and b are created with two attacks:

a *attacks* a $\quad$ b *attacks* a.

Let us now consider the whole logic program consisting of the four rules $r_1, \dots r_4$:

$r1 : a \leftarrow \textit{not}\ a, \textit{not}\ b;$ $\quad$ $r2 : c \leftarrow \textit{not}\ d;$

$r3 : d \leftarrow \textit{not}\ c;$ $\quad$ $r4 : b \leftarrow \textit{not}\ b, \textit{not}\ a, \textit{not}\ c.$

With the encoding of [CSAD15], four arguments are created (one per rule) with the following attacks:

A *attacks* A; $\quad$ B *attacks* B; $\quad$ A *attacks* B;

B *attacks* A; $\quad$ C *attacks* B; $\quad$ D; D *attacks* C.

With the REBAF approach, four prima-facie arguments a, b, c, d are created (one per atom). It can be checked that the obtained AF is the same as the one obtained above with the encoding of [CSAD15], that is the framework pictured in Example 9.2.3.

Argumentation Frameworks with Higher Level Attacks and Logic Programs by Gabbay In [Gab09b] 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.

[Gab09b] 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, [Gab13] 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).

9.5 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 rely on other kinds of translation and then use logic programming for computational issues.

ASPARTIX (Answer Set Programming Argumentation Reasoning Tool) supports reasoning in AFs using the ASP formalism (see [DGW] 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).

[DDLN10] 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 [LN] for the description of the ASP solver ASPeRiX and its application to the credulous acceptance problem in a CAF).

[PLJ17] 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 [CGV14] (see Section 8.3).

10 Argumentation Frameworks and Modal Logic

As outlined in [CG09], 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 $t \in h(q)$
- $t \vDash (A \wedge B), (\neg A), (A \vee B), (A \to B)$ 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
- $\Box(A \to B) \to (\Box A \to \Box B)$
- If $\vDash A$ then $\vDash \Box A$

10.2 The approach by Grossi

In [Gro10] 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.[22] 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 $[\![\phi]\!]$ [23] is conflict-free.
- $[\forall](\phi \rightarrow [\leftarrow]\langle\leftarrow\rangle\psi)$ encodes that $[\![\phi]\!]$ is acceptable wrt $[\![\psi]\!]$.
- $[\forall](\phi \leftrightarrow [\leftarrow]\neg\phi)$ encodes that $[\![\phi]\!]$ is stable.

[22] That is, the codomain of I is the powerset of $\mathcal{A}$.
[23] $[\![\phi]\!]$ denotes the set of arguments in $\mathcal{G}$ that satisfy ϕ.

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:

$$[\forall]\big((1 \leftrightarrow [\leftarrow]0) \wedge (0 \leftrightarrow \langle\leftarrow\rangle 1) \wedge Label\big)$$

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.

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

Furthermore, [Gro10] 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 $a \vDash \phi$ holds in a given model (AF, I) while an opponent tries to disprove it.

Note that an additional modal machinery including a least fixpoint 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 [CG09] is close to the approach of [Gro10]. 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:

$$\begin{array}{l} t \in I(q_0) \text{ (or } t \vDash q_0) \text{ iff } \ell(t) = \texttt{out}; \\ t \in I(q_1) \text{ (or } t \vDash q_1) \text{ iff } \ell(t) = \texttt{in}; \\ t \in I(q_?) \text{ (or } t \vDash q_?) \text{ iff } \ell(t) = \texttt{und}. \end{array}$$

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

Note that the modality $\Box$ corresponds to the modality $[\leftarrow]$ of [Gro10]. However there is no modality corresponding to $[\forall]$.

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

$\Box q_0 \to q_1$	(if all attackers of t are *out* then t is *in*)
$\Diamond q_1 \to q_0$	(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.

- 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 [VBG$^+$12] 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 [CG09] and [Gro10] is that these works describe semantics in the modal language, whereas [VBG$^+$12] considers semantics as primitives of the language.

The approach of [VBG$^+$12] 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 $[\forall]$ in [Gro10]).

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 [CG09] 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:

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

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 [GG16]. The idea is to use intuitionistic negation to model an attack. The intuitionistic models of the obtained theory characterize the complete extensions.
More recently, [FF18] 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 [FF18], intuitionistic logic is used for translating an AF or an EBAF.

Translation of an AF The translation presented in [FF18] 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:

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

[FF18] 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:

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

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, [FF18] proves that there is a one-to-one correspondence between the stable extensions of an EBAF (defined in [CFFLS18]) and the equilibrium models of the theory $C(EBAF)$.

12 Analysis

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

12.1 Comparison criteria

The first question is: "Is argumentation used for doing logic?" or "Is logic used for doing argumentation?". So, two kinds of approaches can be identified:

- Some approaches (most of the rewiewed ones) are "from argumentation to logic" (they use logic for doing argumentation);
- Some other approaches are "from logic to argumentation" (they give an argumentative meaning to a logic program).

The second kind of approaches being very specific (see [Dun95, WCG09, CSAD15]), we focus on the first kind in the following.

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:

   - 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 ouput 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

Input graph	Approaches
Case of an input graph that represents an AF	
Dung	[CFLS17], [GG15], [DHP14], [CMDM06], [BDH14], [BD04], [DJWW12], [CG09], [AC12], [AC13], [DWW14], [DWW15], [CDGV14], [VBG$^+$12], [Dun95], [CNO09], [ONS13], [CSAD15], [ON17], [SR17], [Gro10], [Gab09b], [DBCL16]
Bipolar	[FF18]
Recursive	[CLS18a], [Gab09b]
Weighted	none
With preference	[DGLW15]
With collective interactions	[FF18], [DBCL16]
On a universe	[DBCL16]
Case of an input graph that does not represent an AF	
Dependence graph	ADF approach [BES$^+$17, BES$^+$18]

Other input	Approaches
Acceptance conditions	• (on each argument) ADF approach [BES$^+$17, BES$^+$18]: depending on the chosen acceptance conditions, the dependence graph can represent any type of AF, from Dung AFs to AFs with collective interactions • (on the AF) Constrained AF [CMDM06]: the acceptance condition is used only for removing some extensions
Requirements	On the AF or the semantics [VBG$^+$12]: 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? [CG09], [Gro10], [BDH14], [BD04], [CDGV14], [DJWW12]

Table 1: The first comparison criterion: the input

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.

   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

Aim	Approaches
Translation of an AF	[CLS18a], [CFLS17], [GG15], [DHP14], [DGLW15], [DBCL16], [CNO09], [ONS13], [CSAD15]
Encoding of semantics	
Case of extension-based semantics	
	[BDH14], [BD04], [DJWW12], [CG09], [CLS18a], [CFLS17], [GG15], [DHP14], [DGLW15], [DBCL16], [VBG$^+$12], [Dun95], [CNO09], [ONS13], [CSAD15], [ON17], [SR17], [Nou13], [Gab09b], [FF18]
Case of labelling-based semantics	
	[CG09], [CDGV14], [AC12], [AC13], [DWW14], [DWW15], [CG09], [Gro10]

Table 2: The second comparison criterion: the aim

Used logics	Approaches
Propositional	[BES$^+$17], [BES$^+$18], [CMDM06], [DHP14], [DGLW15], [GG15], [BDH14], [BD04], [DJWW12]
First-order (with finite domains)	[CLS18a], [CFLS17], [CG09], [CDGV14], [DBCL16], [VBG$^+$12]
QBF	[AC12], [AC13], [DWW14], [DWW15]
Logic programming	[Dun95], [CNO09], [ONS13], [CSAD15], [ON17], [SR17], [Nou13], [Gab09b]
Modal	[CG09], [Gro10], [VBG$^+$12]
Intuitionistic	[FF18]

Table 3: The third comparison criterion: the used logic

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 [LLM15]). Table 4 synthetizes all these implementations.

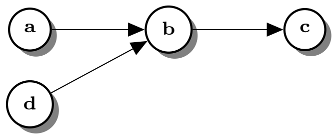
Some existing implementations	Approaches
none	[CMDM06], [VBG$^+$12], [GG15], [DBCL16], [CG09], [Gro10], [FF18]
DIAMOND, UNREAL, GrappaVis	[BES$^+$17], [BES$^+$18]
GRAFIX	[CLS18a], [CFLS17]
SESAME, CoQuiAAS, ...	[BDH14], [BDHL16], [WWW13], [LLM15], [LNJ18], [BD04]
CEGARTIX	[DJWW12]
QADF	[BES$^+$17], [BES$^+$18], [DWW14], [DWW15]
PrefSAT, ArgSemSAT	[CDGV14], [CGV14]
LabSAT	[BBP15], [BBP15]
ASPARTIX, ASPeRiX, ASP-solvers, ...	[Dun95], [CNO09], [ONS13], [CSAD15], [ON17], [SR17], [EGW10], [Gag10], [DDLN10], [PLJ17]

Table 4: The fourth comparison criterion: the 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:*

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

The language is propositional with the vocabulary $V = \{a, b, c, d\}$. *Then, using the ADF machinery (the computation of the fixpoints 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.*

[CLS18a, CFLS17, CLS18b]: *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:

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

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:

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

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 $\{a, d, c\}$).

[GG15]: *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:*[24]

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

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

[DBCL16]: *See Section 6.1. The language is first-order. Considering that the input AF is also the universe, the encoding produces the formula Φ:*

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

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:

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

[CNO09, ONS13, ON17]: *See Section 9.2. The approach uses logic programming. The AF is encoded by the following logic program:*

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

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

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

[CSAD15]: *See Section 9.2. 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.

[FF18]: *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).*

[BD04]: *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:*

$$\begin{aligned}
&[(a \to \neg b) \wedge (a \leftrightarrow \top)] \wedge \\
&[(d \to \neg b) \wedge (d \leftrightarrow \top)] \wedge \\
&[(b \to \neg c) \wedge (b \leftrightarrow \bot)] \wedge \\
&[(c \to \top) \wedge (c \leftrightarrow (a \vee d))]
\end{aligned}$$

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

[DJWW12]: *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:*

$$\begin{aligned}
&[(\neg x_a \vee \neg x_b) \wedge (\neg x_d \vee \neg x_b) \wedge (\neg x_b \vee \neg x_c)] \wedge \\
&[(\neg x_b) \wedge (x_c \to (x_a \vee x_d))] \wedge \\
&[(y_a \leftrightarrow x_a) \wedge (y_d \leftrightarrow x_d)] \wedge \\
&[y_b \leftrightarrow (x_b \vee x_a \vee x_d)] \wedge \\
&[y_c \leftrightarrow (x_c \vee x_b)] \wedge \\
&[y_a \wedge y_d \wedge ((y_a \wedge y_d) \to y_b)] \wedge \\
&[y_b \to y_c]
\end{aligned}$$

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).

[AC12, AC13]: *See Section 5. The logic uses quantified Boolean formulae (QBF formalism). The formulas can be instantiated on a given AF. For the considered AF, the complete labellings are characterized by the following formula:*

$$\begin{aligned}
&[\texttt{val}(a,t) \wedge \neg\texttt{val}(a,f) \wedge \neg\texttt{val}(a,u)] \wedge \\
&[\texttt{val}(d,t) \wedge \neg\texttt{val}(d,f) \wedge \neg\texttt{val}(d,u)] \wedge \\
&[\texttt{val}(b,t) \to (\texttt{val}(a,f) \wedge \texttt{val}(d,f))] \wedge \\
&[\texttt{val}(b,f) \to (\texttt{val}(a,t) \vee \texttt{val}(d,t))] \wedge \\
&\left[\texttt{val}(b,u) \to \left(\begin{array}{l} \neg(\texttt{val}(a,f) \wedge \texttt{val}(d,f)) \\ \wedge(\neg(\texttt{val}(a,t) \vee \texttt{val}(d,t))) \end{array}\right)\right] \wedge \\
&[\texttt{val}(c,t) \to \texttt{val}(b,f)] \wedge \\
&[\texttt{val}(c,f) \to \texttt{val}(b,t)] \wedge \\
&[\texttt{val}(c,u) \to (\neg\texttt{val}(b,f) \wedge \neg\texttt{val}(b,t))] \wedge \\
&[\neg(a^{\oplus} \wedge a^{\ominus})] \wedge \\
&[\neg(b^{\oplus} \wedge b^{\ominus})] \wedge \\
&[\neg(c^{\oplus} \wedge c^{\ominus})] \wedge \\
&[\neg(d^{\oplus} \wedge d^{\ominus})]
\end{aligned}$$

[CG09] with first-order logic: *See Section 8.1. For the considered AF, the complete labellings are characterized by the following formula:*

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

$$\begin{array}{l}
[\neg(Q_0(a) \wedge Q_1(a)) \wedge \neg(Q_0(a) \wedge Q_?(a)) \wedge \neg(Q_?(a) \wedge Q_1(a))] \wedge \\
[\neg(Q_0(b) \wedge Q_1(b)) \wedge \neg(Q_0(b) \wedge Q_?(b)) \wedge \neg(Q_?(b) \wedge Q_1(b))] \wedge \\
[\neg(Q_0(c) \wedge Q_1(c)) \wedge \neg(Q_0(c) \wedge Q_?(c)) \wedge \neg(Q_?(c) \wedge Q_1(c))] \wedge \\
[\neg(Q_0(d) \wedge Q_1(d)) \wedge \neg(Q_0(d) \wedge Q_?(d)) \wedge \neg(Q_?(d) \wedge Q_1(d))] \wedge \\
[(Q_0(a) \rightarrow Q_1(b)) \wedge (Q_0(d) \rightarrow Q_1(b)) \wedge Q_1(a) \wedge Q_1(d)] \wedge \\
[Q_0(b) \rightarrow Q_1(c)] \wedge \\
[((Q_1(a) \vee Q_1(d)) \rightarrow Q_0(b)) \wedge (Q_1(b) \rightarrow Q_0(c))] \wedge \\
[((Q_0(b) \vee Q_?(b)) \wedge Q_?(b)) \rightarrow Q_?(c)] \wedge \\
\left[\left(\begin{array}{l} (Q_0(a) \vee Q_?(a)) \\ \wedge(Q_0(d) \vee Q_?(d)) \\ \wedge(Q_?(a) \vee Q_?(d)) \end{array} \right) \rightarrow Q_?(b) \right]
\end{array}$$

[CG09] with modal logic: *See Section 10.3. The modal formula characterizing the complete extensions is quite simple. The difficulty lies in computational issues.*

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

[Gro10]: *See Section 10.2. The logic is a modal logic. The modal formula characterizing the complete extensions is the following one:*

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

Here again, the difficulty lies in computational issues.

References

[AC12] Ofer Arieli and Martin W. A. Caminada. A general QBF-based formalization of abstract argumentation theory. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Proceedings of the 4th Conference on Computational Models of Argument (COMMA'2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 105–116, Vienna, Austria, September 2012. IOS Press.

[AC13] Ofer Arieli and Martin W. A. Caminada. A QBF-based formalization of abstract argumentation semantics. *J. Applied Logic*, 11(2):229–252, 2013.

[BBD$^+$] Philippe Besnard, Joseph Boudou, Sylvie Doutre, Van Hieu Ho, and Dominique Longin. The SESAME website. `https://www.irit.fr/SESAME/`.

[BBP15] Christoph Beierle, Florian Brons, and Nico 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, Dresden, Germany, September 21-25, 2015, Proceedings*, pages 241–248, 2015.

[BCDLS13] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr Bannay, and Marie-Christine Lagasquie-Schiex. Characterizing change in abstract argumentation systems. In Edouardo Fermé, Dov Gabbay, and Guillermo Simari, editors, *Trends in Belief Revision and Argumentation Dynamics*, volume 48 of *Studies in Logic*, pages 75–102. College Publications, http://www.collegepublications.co.uk/, 2013.

[BCG11] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26(4):365–410, 2011.

[BCGG11] Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Giovanni Guida. AFRA: Argumentation framework with recursive attacks. *International Journal of Approximate Reasoning*, 52:19–37, 2011.

[BD04] Philippe Besnard and Sylvie Doutre. Checking the acceptability of a set of arguments. In James P. Delgrande and Torsten Schaub, editors, *10th International Workshop on Non-Monotonic Reasoning (NMR'04)*, pages 59–64, Whistler, Canada, June 2004.

[BDH14] Philippe Besnard, Sylvie Doutre, and Andreas Herzig. Encoding Argument Graphs in Logic (regular paper). In Anne Laurent, Olivier Strauss, Bernadette Bouchon-Meunier, and Ronald R. Yager, editors, *International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU), Montpellier, France, 15/07/2014-19/07/2014*, volume 443 of *Communications in Computer and Information Science*, pages 345–354, http://www.springerlink.com, 2014. Springer.

[BDHL16] Philippe Besnard, Sylvie Doutre, Van Hieu Ho, and Dominique Longin. SESAME - A System for Specifying Semantics in Abstract Argumentation (regular paper). In Matthias Thimm, Federico Cerutti, Hannes Strass, and Mauro Vallati, editors, *International Workshop on Systems and Algorithms for Formal Argumentation (SAFA), Potsdam, Germany, 13/09/2016*, volume 1672, pages 40–51, http://CEUR-WS.org, septembre 2016. CEUR Workshop Proceedings.

[BES$^+$17] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes P. Wallner, and Stefan Woltran. Abstract dialectical frameworks. an overview. *The IfCoLog Journal of Logics and their Applications*, 4(8):2263–2318, 2017.

[BES$^+$18] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes P. Wallner, and Stefan Woltran. Abstract dialectical frameworks. In Pietro Baroni, Dov Gabbay, Massimiliano Giacomin, and Leendert van der Torre, editors, *Handbook of Formal Argumentation*, pages 237–286 (Chapter 5). College Publications, 2018.

[BGvdTV10] 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*, pages 111–122, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[BGW05] Howard Barringer, Dov M. Gabbay, and John Woods. Temporal dynamics of support and attack networks : From argumentation to zoology. In Dieter Hutter and Werner Stephan, editors, *Mechanizing Mathematical Reasoning. LNAI 2605*, pages 59–98. Springer Verlag, 2005.

[BKRv13] Richard Booth, Souhila Kaci, Tjitze Rienstra, and Leendert van der Torre. A logical theory about dynamics in abstract argumentation. In Weiru Liu, V.S. Subrahmanian, and Jef Wijsen, editors, *Scalable Uncertainty Management*, volume 8078 of *Lecture Notes in Computer Science*, pages 148–161. Springer Berlin Heidelberg, 2013.

[BWa] Gerhard Brewka and Stefan Woltran. The ADF Project website. `http://www.dbai.tuwien.ac.at/research/project/adf/`.

[BWb] Gerhard Brewka and Stefan Woltran. The GRAPPA Project website. `http://www.dbai.tuwien.ac.at/proj/grappa/`.

[CDGV14] Federico Cerutti, Paul E. Dunne, Massimiliano Giacomin, and Mauro Vallati. Computing preferred extensions in abstract argumentation: A sat-based approach. In E. Black, S. Modgil, and N. Oren, editors, *Theory and Applications of Formal Argumentation, TAFA 2013*, volume 8306 of *LNAI*, pages 176–193. Springer-Verlag Berlin Heidelberg, 2014.

[CDLS10] Claudette Cayrol, Florence Dupin de Saint-Cyr Bannay, and Marie-Christine Lagasquie-Schiex. Change in abstract argumentation frameworks: Adding an argument. *Journal of Artificial Intelligence Research*, 38:49–84, 2010.

[CFFLS17] Claudette Cayrol, Jorge Fandinno, Luis Fariñas del Cerro, and Marie-Christine Lagasquie-Schiex. Valid attacks in argumentation frameworks with recursive attacks. In *Proc. of Thirteenth International Symposium on Commonsense Reasoning*, volume 2052. CEUR Workshop Proceedings, 2017.

[CFFLS18] Claudette Cayrol, Jorge Fandinno, Luis Fariñas del Cerro, and Marie-Christine Lagasquie-Schiex. Argumentation frameworks with recursive attacks and evidence-based support. In F. Ferrarotti and S. Woltran, editors, *Proc. of Tenth International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, volume LNCS 10833, pages 150–169. Springer-Verlag, 2018.

[CFLS17] Claudette Cayrol, Luis Fariñas del Cerro, and Marie-Christine Lagasquie-Schiex. Logical encodings of interactions in an argumentation graph with recursive attacks. Technical Report RR–2017-08–FR, IRIT, 2017.

[CG09] Martin W. A. Caminada and Dov M. Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2-3):109–145, 2009.

[CGGS14] Andrea Cohen, Sebastian Gottifredi, Alejandro J. García, and Guillermo R. Simari. A survey of different approaches to support in argumentation systems. *The Knowledge Engineering Review*, 29:513–550, 11 2014.

[CGGS15] Andrea Cohen, Sebastian Gottifredi, Alejandro J García, and Guillermo R Simari. An approach to abstract argumentation with recursive attack and support. *Journal of Applied Logic*, 13(4):509–533, 2015.

[CGV14] Federico Cerutti, Massimiliano Giacomin, and Mauro Vallati. Argsemsat: Solving argumentation problems using SAT. In *Computational Models of Argument - Proceedings of COMMA*, pages 455–456, 2014.

[CLS] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. The Grafix website. `http://www.irit.fr/grafix`.

[CLS05] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Gradual valuation for bipolar argumentation frameworks. In Luis Godo, editor, *Proc. of the eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU) – LNAI 3571*, pages 366–377, Barcelona, Spain, 2005. Springer-Verlag.

[CLS13] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolarity in argumentation graphs: towards a better understanding. *International Journal of Approximate Reasoning*, 54(7):876–899, 2013.

[CLS18a] Claudette Cayrol and Marie-Christine 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, janvier 2018.

[CLS18b] Claudette Cayrol and Marie-Christine 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.

[CMDM06] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Constrained argumentation frameworks. In *Proc. of KR*, pages 112–122, Lake District, 2006.

[CMKMM14a] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis. On the revision of argumentation systems: Minimal change of argument statuses. In C. Baral and G. De Giacomo, editors, *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 72–81. AAAI Press, 2014.

[CMKMM14b] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis. A translation-based approach for revision of argumentation frameworks. In *Logics in Artificial Intelligence*, pages 397–411. Springer, 2014.

[CNO09] José Luis Carballido, Juan Carlos Nieves, and Mauricio Osorio. Inferring preferred extensions by pstable semantics. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 13(41):38–53, 2009.

[CSAD15] Martin Caminada, Samy Sá, João Alcântara, and Wolfgang Dvorák. On the equivalence between logic programming semantics and argumentation semantics. *Int. J. Approx. Reasoning*, 58:87–111, 2015.

[DBCL16] Florence Dupin de Saint-Cyr, Pierre Bisquert, Claudette Cayrol, and Marie-Christine Lagasquie-Schiex. Argumentation update in YALLA (yet another logic language for argumentation). *Int. J. Approx. Reasoning*, 75:57–92, 2016.

[DDLN10] Caroline Devred, Sylvie Doutre, Claire Lefèvre, and Pascal Nicolas. Dialectical proofs for constrained argumentation. In *Computational Models of Argument - Proceedings of COMMA*, pages 159–170, 2010.

[DGLW15] Wolfgang Dvořák, Sarah Alice Gaggl, Thomas Linsbichler, and Johannes Peter Wallner. Reduction-based approaches to implement Modgil's extended argumentation frameworks. In Thomas Eiter, Hannes Strass, Miroslaw Truszczynski, and Stefan Woltran, editors, *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of his 60th Birthday*, volume 9060 of *Lecture Notes in Computer Science*, pages 249–264. Springer, 2015.

[DGW] Wolfgang Dvořák, Sarah Alice Gaggl, and Stefan Woltran. The ASPARTIX website. `https://www.dbai.tuwien.ac.at/research/argumentation/aspartix/`.

[DHL+15] Martin Diller, Adrian Haret, Thomas Linsbichler, Stefan Rümmele, and Stefan 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)*, pages 2926–2932, 2015.

[DHP14] Sylvie Doutre, Andreas Herzig, and Laurent Perrussel. A dynamic logic framework for abstract argumentation. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'2014)*, Vienna, Austria, July 2014. AAAI Press.

[DJWW] Wolfgang Dvořák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. The CEGARTIX website. `https://www.dbai.tuwien.ac.at/proj/argumentation/cegartix/`.

[DJWW12] Wolfgang Dvořák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. CEGARTIX: a SAT-based argumentation system. In *Proc. of Pragmatics of SAT, workshop of SAT conference*, 2012.

[Dun95] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[DWW14] Martin Diller, Johannes Peter Wallner, and Stefan Woltran. Reasoning in abstract dialectical frameworks using quantified boolean formulas. In Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti, editors, *Proceedings of the 5th Conference on Computational Models of Argument (COMMA'2014)*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 241–252, Pitlochry, Scottish Highlands, UK, September 2014. Held in Atholl Palace Hotel.

[DWW15] Martin Diller, Johannes Peter Wallner, and Stefan Woltran. Reasoning in abstract dialectical frameworks using quantified boolean formulas. *Argument & Computation*, 6(2):149–177, 2015.

[EGW10] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument & Computation*, 1(2):147–177, 2010.

[FF18] Jorge Fandinno and Luis Fariñas del Cerro. Constructive logic covers argumentation and logic programming. In F. Toni M. Thielscher and F. Wolter, editors, *Proc. of Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2018)*, pages 128–137. AAAI Press, 2018.

[Gab09a] Dov M. Gabbay. Fibring argumentation frames. *Studia Logica*, 93:231–295, 2009.

[Gab09b] Dov M. Gabbay. Semantics for higher level attacks in extended argumentation frames part 1: Overview. *Studia Logica*, 93(2-3):357–381, 2009.

[Gab13] Dov M. Gabbay. *Meta-logical Investigations in Argumentation Networks.* Number 44 in Studies in logic. College Publications, 2013.

[Gag10] Sarah Alice Gaggl. Towards a general argumentation system based on answer-set programming. In Manuel V. Hermenegildo and Torsten Schaub, editors, *Technical Communications of the 26th International Conference on Logic Programming (ICLP 2010)*, volume 7 of *LIPIcs*, pages 265–269. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, July 2010.

[GG15] Dov M. Gabbay and Michael Gabbay. The attack as strong negation, part I. *Logic Journal of the IGPL*, 23(6):881–941, 2015.

[GG16] Dov M. Gabbay and Michael Gabbay. The attack as intuitionistic negation. *Logic Journal of the IGPL*, 24(5):807–837, 2016.

[Gro10] Davide Grossi. On the logic of argumentation theory. In *Proc. of AAMAS*, pages 409–416, 2010.

[KP01] Nikos Karacapilidis and Dimitris Papadias. Computer supported argumentation and collaborative decision making: the HERMES system. *Information systems*, 26(4):259–277, 2001.

[LLM15] Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly. Coquiaas: A constraint-based quick abstract argumentation solver. In *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI*, pages 928–935, 2015.

[LN] Claire Lefèvre and Pascal Nicolas. The ASPeRiX website. `http://www.info.univ-angers.fr/pub/claire/asperix/Argumentation/`.

[LNJ18] Tuomo Lehtonen, Andreas Niskanen, and Matti Järvisalo. Sat-based approaches to adjusting, repairing, and computing largest extensions of argumentation frameworks. In *Computational Models of Argument - Proceedings of COMMA*, pages 193–204, 2018.

[Mod09] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173:901–934, 2009.

[Nou13] Farid Nouioua. AFs with necessities: further semantics and labelling characterization. In *Proc. of the International Conference on Scalable Uncertainty Management (SUM)*, pages 120–133, 2013.

[NR10] Farid Nouioua and Vincent Risch. Bipolar argumentation frameworks with specialized supports. In *Proc. of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 215–218. IEEE Computer Society, 2010.

[NR11] Farid Nouioua and Vincent Risch. Argumentation frameworks with necessities. In *Proc. of the International Conference on Scalable Uncertainty Management (SUM)*, pages 163–176. Springer-Verlag, 2011.

[ON08] Nir Oren and Timothy J. Norman. Semantics for evidence-based argumentation. In Philippe Besnard, Sylvie Doutre, and Anthony Hunter, editors, *Proc of the* 2^{nd} *international conference* *on Computational Models of Argument (COMMA)*, pages 276–284. IOS Press, 2008.

[ON17] Mauricio Osorio and Juan Carlos Nieves. Range-based argumentation semantics as two-valued models. *TPLP*, 17(1):75–90, 2017.

[ONS13] Mauricio Osorio, Juan Carlos Nieves, and Alejandro 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*, pages 81–88, 2013.

[ORL10] Nir Oren, Chris Reed, and Michael Luck. Moving between argumentation frameworks. In *Proceeding of the conference on Computational Models of Argument (COMMA)*, pages 379–390, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[PLJ17] Fuan Pu, Guiming Luo, and Zhou Jiang. Encoding argumentation semantics by boolean algebra. *IEICE Transactions*, 100-D(4):838–848, 2017.

[PO14] S. Polberg and N. Oren. Revisiting support in abstract argumentation systems. Technical report, TU Wien, Institut for Informatics, 2014.

[Pol17] Sylwia 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.

[SR17] Chichi Sakama and Tjitze Rienstra. Representing argumentation frameworks in answer set programming. *Fundamenta Informaticae*, 155(3):261–292, 2017.

[VBG$^+$12] Serena Villata, Guido Boella, Dov M. Gabbay, Leendert van der Torre, and Joris Hulstijn. A logic of argumentation for specification and verification of abstract argumentation frameworks. *Annals of Mathematics and Artificial Intelligence*, 66(1-4):199–230, 2012.

[VBGvdT12] Serena Villata, Guido Boella, Dov M. Gabbay, and Leendert van der Torre. Modelling defeasible and prioritized support in bipolar argumentation. *Annals of Mathematics and Artificial Intelligence*, 66(1-4):163–197, 2012.

[Ver03] Bart Verheij. Deflog: on the logical interpretation of prima facie justified assumptions. *Journal of Logic in Computation*, 13:319–346, 2003.

[WCG09] Yining Wu, Martin Caminada, and Dov 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):383–403, 2009.

[WD12] Michal Walicki and Sjur Dyrkolbotn. Finding kernels or solving SAT. *Discrete Algorithms*, 10:146–164, 2012.

[WWW13] Johannes Peter Wallner, Georg Weissenbacher, and Stefan Woltran. Advanced SAT techniques for abstract argumentation. In *Computational Logic in Multi-Agent Systems - 14th International Workshop, CLIMA XIV, Corunna, Spain, September 16-18, 2013. Proceedings*, pages 138–154, 2013.

Figures

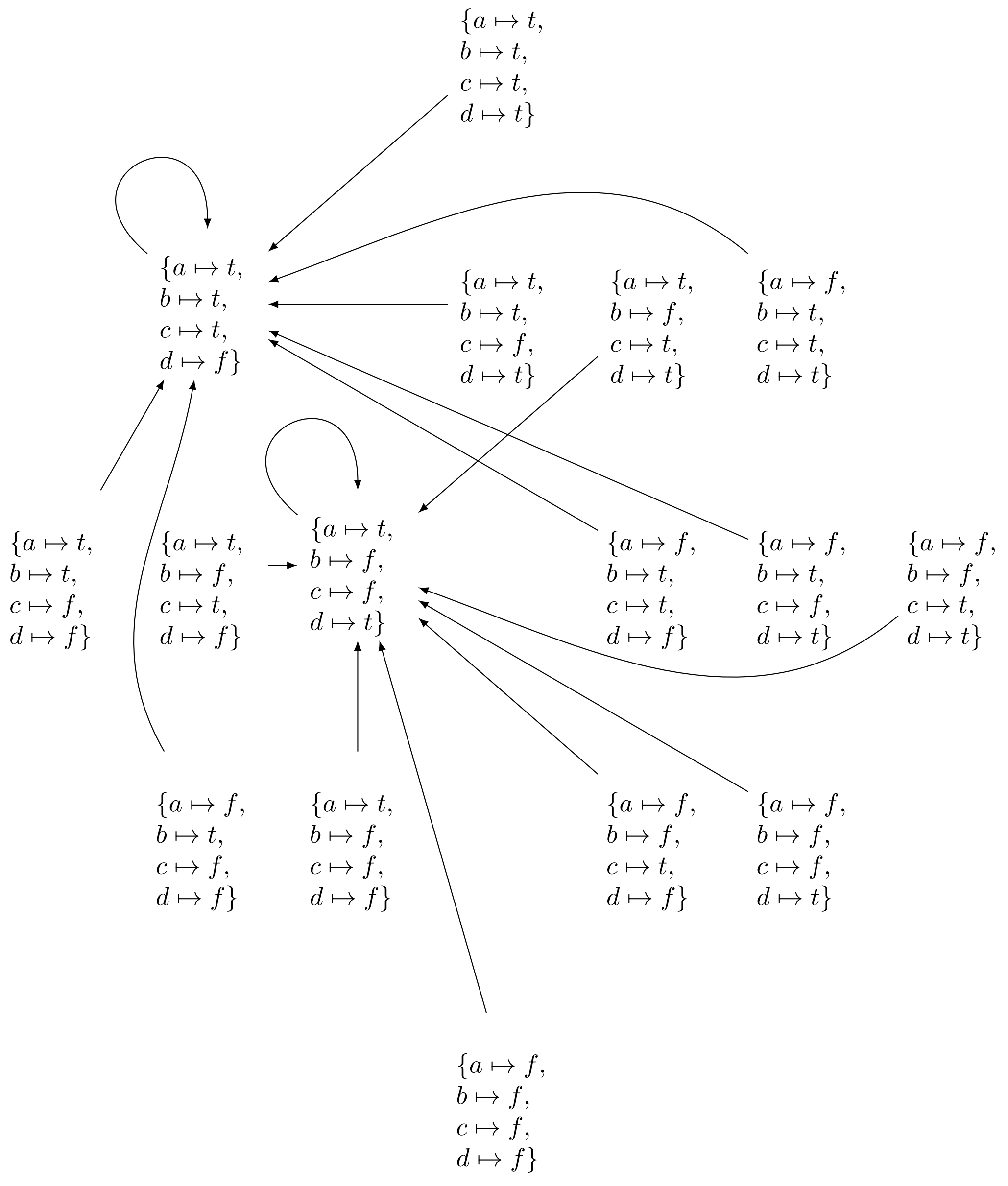

$\{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto t\}$
$\{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto f\}$
$\{a \mapsto t, b \mapsto t, c \mapsto f, d \mapsto t\}$
$\{a \mapsto t, b \mapsto f, c \mapsto t, d \mapsto t\}$
$\{a \mapsto f, b \mapsto t, c \mapsto t, d \mapsto t\}$
$\{a \mapsto t, b \mapsto f, c \mapsto f, d \mapsto t\}$
$\{a \mapsto t, b \mapsto t, c \mapsto f, d \mapsto f\}$
$\{a \mapsto t, b \mapsto f, c \mapsto t, d \mapsto f\}$
$\{a \mapsto f, b \mapsto t, c \mapsto t, d \mapsto f\}$
$\{a \mapsto f, b \mapsto t, c \mapsto f, d \mapsto t\}$
$\{a \mapsto f, b \mapsto f, c \mapsto t, d \mapsto t\}$
$\{a \mapsto f, b \mapsto t, c \mapsto f, d \mapsto f\}$
$\{a \mapsto t, b \mapsto f, c \mapsto f, d \mapsto f\}$
$\{a \mapsto f, b \mapsto f, c \mapsto t, d \mapsto f\}$
$\{a \mapsto f, b \mapsto f, c \mapsto f, d \mapsto t\}$
$\{a \mapsto f, b \mapsto f, c \mapsto f, d \mapsto f\}$

Figure 3: ADF approach, Example 3.2.1: Set of two-valued interpretations and G_D operator

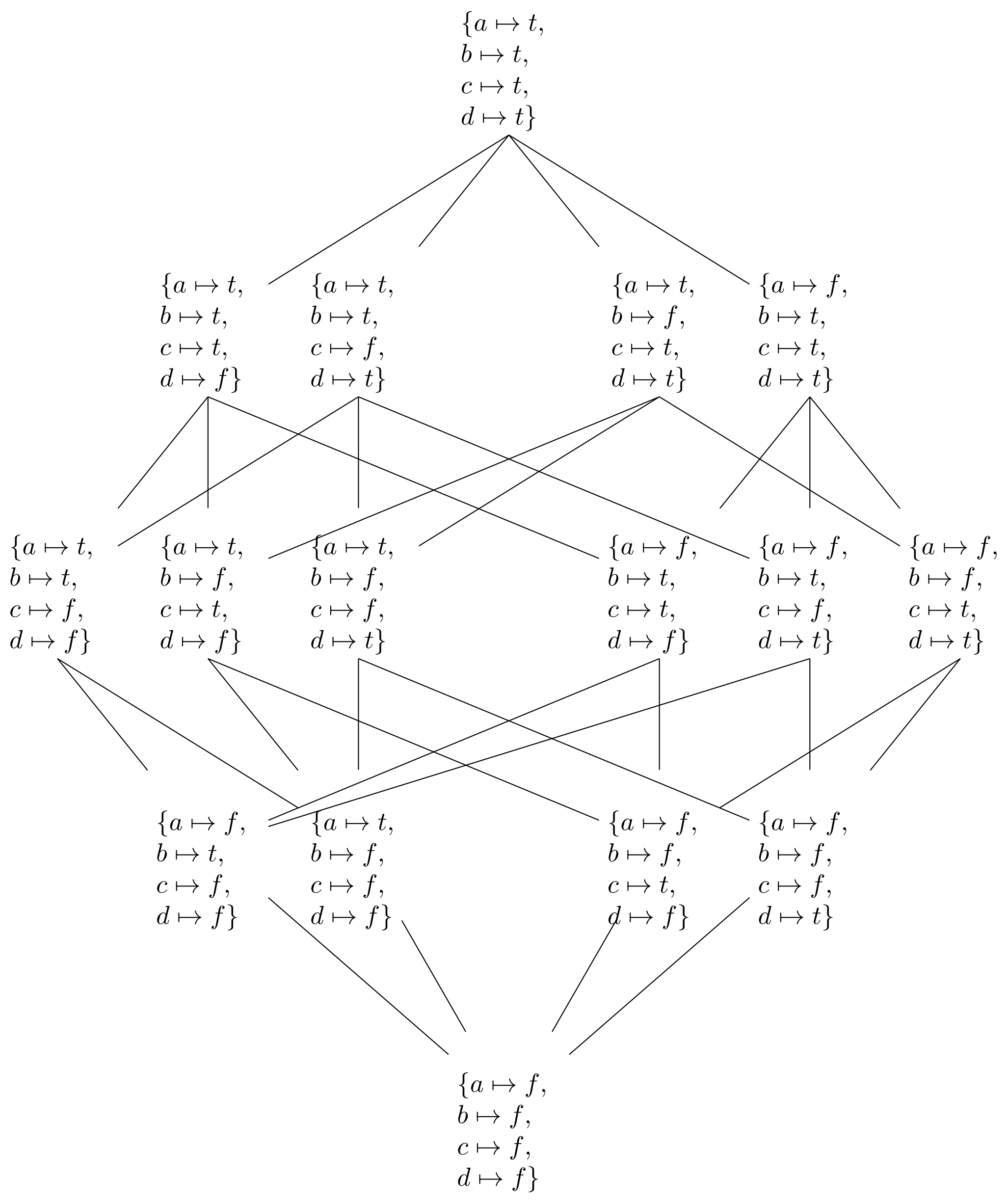

{a ↦ t, b ↦ t, c ↦ t, d ↦ t}
{a ↦ t, b ↦ t, c ↦ t, d ↦ f}
{a ↦ t, b ↦ t, c ↦ f, d ↦ t}
{a ↦ t, b ↦ f, c ↦ t, d ↦ t}
{a ↦ f, b ↦ t, c ↦ t, d ↦ t}
{a ↦ t, b ↦ t, c ↦ f, d ↦ f}
{a ↦ t, b ↦ f, c ↦ t, d ↦ f}
{a ↦ t, b ↦ f, c ↦ f, d ↦ t}
{a ↦ f, b ↦ t, c ↦ t, d ↦ f}
{a ↦ f, b ↦ t, c ↦ f, d ↦ t}
{a ↦ f, b ↦ f, c ↦ t, d ↦ t}
{a ↦ f, b ↦ t, c ↦ f, d ↦ f}
{a ↦ t, b ↦ f, c ↦ f, d ↦ f}
{a ↦ f, b ↦ f, c ↦ t, d ↦ f}
{a ↦ f, b ↦ f, c ↦ f, d ↦ t}
{a ↦ f, b ↦ f, c ↦ f, d ↦ f}

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

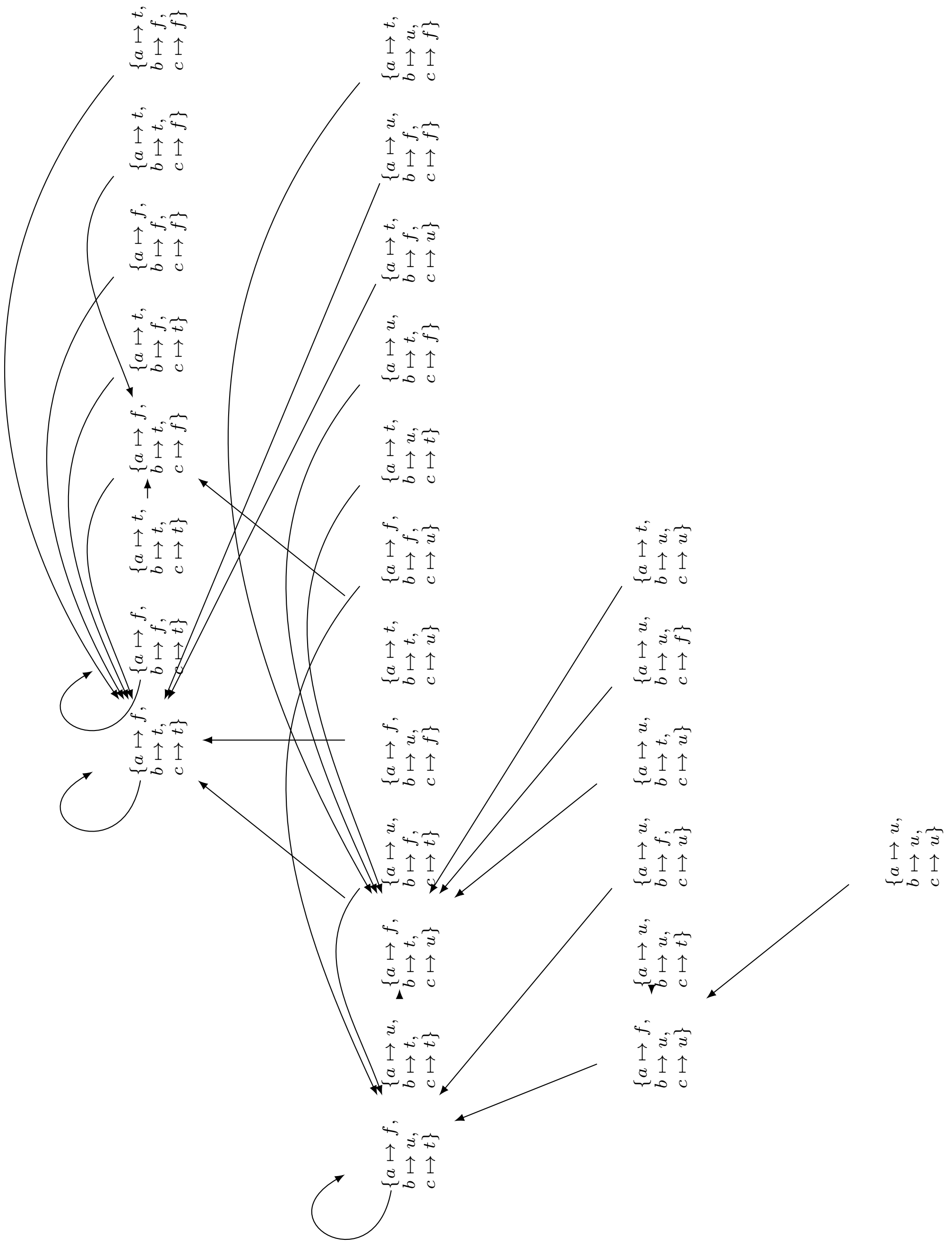


Figure 5: ADF approach, Example 3.2.2: Set of three-valued interpretations and Γ_D operator

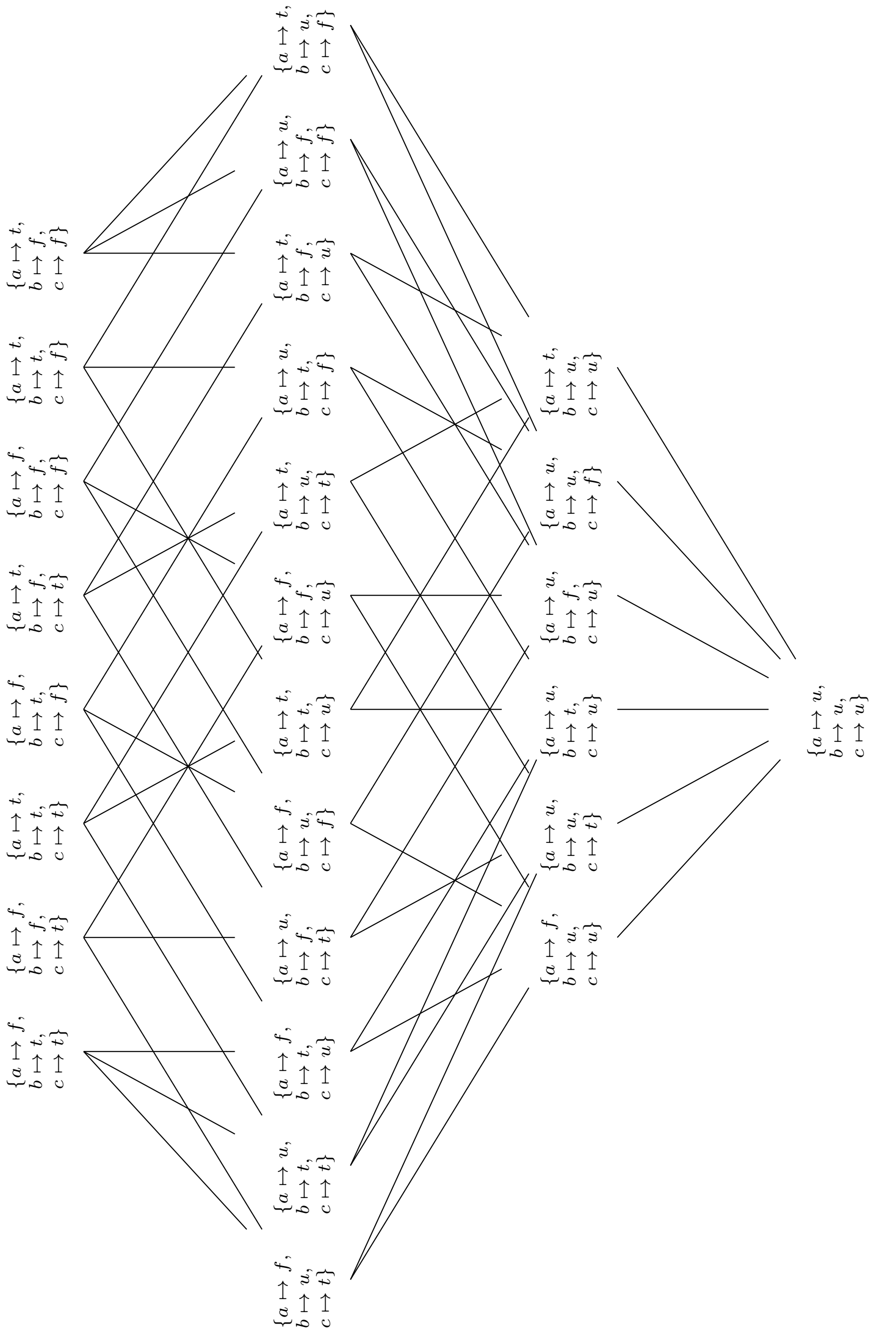

$\{a \mapsto t, b \mapsto f, c \mapsto f\}$
$\{a \mapsto t, b \mapsto t, c \mapsto f\}$
$\{a \mapsto f, b \mapsto f, c \mapsto f\}$
$\{a \mapsto t, b \mapsto f, c \mapsto t\}$
$\{a \mapsto f, b \mapsto t, c \mapsto f\}$
$\{a \mapsto t, b \mapsto t, c \mapsto t\}$
$\{a \mapsto f, b \mapsto f, c \mapsto t\}$
$\{a \mapsto f, b \mapsto t, c \mapsto t\}$
$\{a \mapsto t, b \mapsto u, c \mapsto f\}$
$\{a \mapsto u, b \mapsto f, c \mapsto f\}$
$\{a \mapsto t, b \mapsto f, c \mapsto u\}$
$\{a \mapsto u, b \mapsto t, c \mapsto f\}$
$\{a \mapsto t, b \mapsto u, c \mapsto t\}$
$\{a \mapsto f, b \mapsto f, c \mapsto u\}$
$\{a \mapsto t, b \mapsto t, c \mapsto u\}$
$\{a \mapsto f, b \mapsto u, c \mapsto f\}$
$\{a \mapsto u, b \mapsto f, c \mapsto t\}$
$\{a \mapsto f, b \mapsto t, c \mapsto u\}$
$\{a \mapsto u, b \mapsto t, c \mapsto t\}$
$\{a \mapsto f, b \mapsto u, c \mapsto t\}$
$\{a \mapsto t, b \mapsto u, c \mapsto u\}$
$\{a \mapsto u, b \mapsto u, c \mapsto f\}$
$\{a \mapsto u, b \mapsto f, c \mapsto u\}$
$\{a \mapsto u, b \mapsto t, c \mapsto u\}$
$\{a \mapsto u, b \mapsto u, c \mapsto t\}$
$\{a \mapsto f, b \mapsto u, c \mapsto u\}$
$\{a \mapsto u, b \mapsto u, c \mapsto u\}$

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