ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-153

GRAPPA: A Semantical Framework for Graph-Based Argument Processing[1]

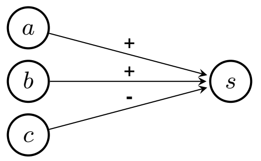
Gerhard Brewka[2] and **Stefan Woltran**[3]

Abstract. Graphical models are widely used in argumentation to visualize relationships among propositions or arguments. The intuitive meaning of the links in the graphs is typically expressed using labels of various kinds. In this paper we introduce a general semantical framework for assigning a precise meaning to labelled argument graphs which makes them suitable for automatic evaluation. Our approach rests on the notion of explicit acceptance conditions, as first studied in Abstract Dialectical Frameworks (ADFs). The acceptance conditions used here are functions from multisets of labels to truth values. We define various Dung style semantics for argument graphs. We also introduce a pattern language for specifying acceptance functions. Moreover, we show how argument graphs can be compiled to ADFs, thus providing an automatic evaluation tool via existing ADF implementations. Finally, we also discuss complexity issues.

1 INTRODUCTION

Graphical models play an important role in many scientific areas including AI (just think of Bayes nets for probabilistic reasoning). Graphical representations seem particular useful - and are indeed widely used - in argumentation, one of the highly active subfields of AI. Here, it is common to lay out the structure of a particular argumentation scenario using graphs of various kinds. In fact, argument mapping is a field of its own, and there are even commercial systems around supporting graphical representations of argumentation scenarios. A prominent example is Rationale (`http://rationale.austhink.com/`), an educational tool based on argument mapping. Systems like Rationale allow the user to create graphs of various kinds, but do not come with a semantics of links and nodes. In other words, they are tools for visualization, but not for semantic evaluation of the graphs.

The framework we are developing in this paper tries to fill this gap. Our framework allows not only to visualize argument structures, but also to assign a semantics to a wide range of graphs. More precisely, we consider arbitrary labelled graphs. Each link in the graph carries a label. There is no restriction as to the labels used, they can be arbitrary symbols including numbers. As an example, take a simple argument graph containing links labelled with + or -, representing support and attack, respectively. Now assume a situation depicted as follows:



Whether s should be accepted or not certainly depends on the acceptance status of its parents. However, even if the parents' status is known, the intuitive meaning of the labels is still not sufficient to decide s. Let's call a link active if its source node is accepted. Different options arise. For instance, we might say s should be accepted iff

- no negative and all positive links are active, or
- no negative and at least one positive link is active, or
- more positive than negative links are active.

The bottom line is: to evaluate an argument graph we not only need labelled links, but also an acceptance condition for each of the nodes. In this paper we introduce a general framework for handling acceptance conditions which are based on the labels of active links. In addition, we provide a formal language called GRAPPA (GRaph-based Argument Processing with Patterns of Acceptance) to express such conditions conveniently. Since they can be defined individually for each node, we end up with graphs where links have a label, taken from some arbitrary set, and nodes come with acceptance conditions represented as patterns in our language.

Explicit acceptance conditions for argumentation were first studied in the context of ADFs, a generalization of Dung frameworks [9], initially proposed in [5] and further developed in [3]. The latter paper redefines the semantics of ADFs based on a so-called characteristic operator, a technique going back to a general operator-based theory of nonmonotonic reasoning developed by [8]. Acceptance conditions for ADFs are propositional formulas with variables ranging over the parents of a node. In the approach developed here we specify acceptance conditions as functions from sets of labels to truth values. Intuitively, we collect the labels of the active links and check whether the set obtained this way satisfies the specified condition. Since the number of occurrences of a particular label obviously may make a difference, we actually have to use multisets of labels.

The rest of the paper is organized as follows. In Sect. 2 we introduce labelled argument graphs (LAGs) and define various Dung style semantics for them. Sect. 3 introduces the GRAPPA pattern language for acceptance conditions. Sect. 4 illustrates the potential of GRAPPA by handling typical graph-based approaches to argumentation. Sect. 5 shows how labelled argument graphs and the GRAPPA approach can be represented as standard ADFs which allows us to use existing tools for the implementation. We also discuss the complexity of GRAPPA. Sect. 6 concludes.

[1] This research has been supported by DFG (projects BR 1817/7-1 and FOR 1513) and FWF (project I1102). The reseach was carried out during the second author's stay at Leipzig University.
[2] Leipzig University, Informatics Institute, Postfach 100920, 04009 Leipzig, Germany, brewka@informatik.uni-leipzig.de
[3] TU Vienna, Institute of Information Systems, Favoritenstraße 9–11, A-1040 Vienna, woltran@dbai.tuwien.ac.at

2 LABELLED ARGUMENT GRAPHS

In this section we will formally introduce labelled argument graphs (LAGs). As mentioned in the Introduction, an essential ingredient for the evaluation of such graphs are acceptance functions based on multisets of labels. For this reason we introduce acceptance functions based on a set L of labels first. A multiset m of labels taken from L can formally be viewed as a function $m : L \to \mathbb{N}$ such that for each $l \in L$, $m(l)$ is the number of occurrences of l in the multiset. As usual, we will often represent multisets using standard set notation, but with multiple occurrences of elements. For instance, $\{+,+,-\}$ is used to denote the multiset m with $m(+) = 2$, $m(-) = 1$, and $m(x) = 0$ for any other elements of L (if there are any).

Definition 1 *Let L be a set of labels. An acceptance function over L (L-acceptance function for short) is a function $c : (L \to \mathbb{N}) \to \{t, f\}$, that is, a function assigning a truth value to a multi-set of labels. The set of all L-acceptance functions is denoted F^L.*

We are now ready to define LAGs:

Definition 2 *A labelled argument graph (LAG) is a tuple $G = (S, E, L, \lambda, \alpha)$ where*

- *S is a set of nodes (statements),*
- *E is a set of edges (dependencies),*
- *L is a set of labels,*
- *$\lambda : E \to L$ assigns labels to edges,*
- *$\alpha : S \to F^L$ assigns L-acceptance-functions to nodes.*

The semantics of LAGs is defined in a similar way as the semantics of ADFs [3], namely by introducing a characteristic operator whose (pre)fixpoints will give us the intended semantics. The operator is based on partial interpretations.[4] Partial interpretations of S assign a truth value from t, f standing for true, respectively false, to *some* of the nodes in S, leaving the truth values of the other nodes open. They thus generalize classical interpretations where the set of open nodes is empty. Partial interpretations can be viewed as representing what is known or assumed about S in a particular situation.

Partial interpretations are conveniently represented as sets of literals containing elements of S evaluated to t unnegated and those evaluated to f negated. Given a partial interpretation v of S, a completion of v is a classical (total) interpretation of S (in other words: a consistent set of literals containing s or $\neg s$ for each $s \in S$) containing v. The set of all completions of v is denoted $[v]_c$.

The intuition behind the operator we define next is as follows. Consider a partial interpretation v over the nodes in S. The operator revises v and produces a new partial interpretation v'. In doing so, it checks which truth values of nodes in S can be justified, based on v. This is done by considering all possible completions of v in the following way: if the acceptance function of a node s evaluates to t under all completions, that is whatever the truth values of the open nodes are, then v' assigns t to s. If the evaluation of the acceptance condition for s yields f for all completions, then the value of s under v' is f. In all other cases the value remains open.

Definition 3 *Let $G = (S, E, L, \lambda, \alpha)$ be an LAG, v a partial interpretation of S. m_s^v, the multiset of active labels of $s \in S$ in G under v, is defined as*

$$m_s^v(l) = |\{(e, s) \in E \mid e \in v, \lambda((e, s)) = l\}|$$

for each $l \in L$.

The characteristic operator Γ_G of G takes a partial interpretation v of S and produces a revised partial interpretation $\Gamma_G(v)$ of S defined as follows: $\Gamma_G(v) = P_G(v) \cup N_G(v)$ with

$$P_G(v) = \{s \mid \alpha(s)(m) = t \text{ for each } m \in \{m_s^{v'} \mid v' \in [v]_c\}\}$$
$$N_G(v) = \{\neg s \mid \alpha(s)(m) = f \text{ for each } m \in \{m_s^{v'} \mid v' \in [v]_c\}\}$$

On the basis of this operator we can now define various semantics for LAGs. Except for the underlying operator, the definitions are exactly those for ADFs introduced in [3] as generalizations of Dung's AF semantics [9].

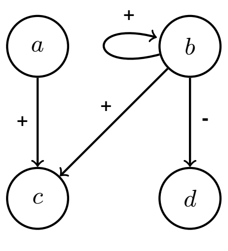
Before introducing the formal definitions, let's describe the motivations underlying the semantics. Consider a classical, total interpretation v which is a fixed point of Γ_G. In this case the assignment of truth values is such that a node is t iff its acceptance function evaluates to t, and f otherwise. We will call such interpretations models. Now consider the least fixed point of Γ_G. Since this operator is easily shown to be $\subseteq$-monotonic, the least fixed point can be computed by iterating on the empty partial interpretation where every node is open. It is easy to see that in each step only nodes receive truth value t, respectively f, for which the respective assignment is beyond any doubt, that is where the assignment must be the right one according to the acceptance function, independently of what the truth value of open nodes may turn out to be.

Admissible interpretations are "safe" in the sense that whatever the truth value of the open nodes is, assignments of values t and f are justified and will be preserved. This is captured by requiring that revising an interpretation v leads to an interpretation containing the information in v. Preferred interpretations are then those admissible interpretations which contain maximal information. Finally, it is natural to consider an interpretation v as complete whenever applying the revision operator reproduces v, in other words whenever v is a fixed point of Γ_G. This leads to the following definitions:

Definition 4 *Let $G = (S, E, L, \lambda, \alpha)$ be an LAG, v a partial interpretation of S. We say*

- *v is a model of G iff v is total and $v = \Gamma_G(v)$,*
- *v is grounded in G iff v is the least fixed point of Γ_G,*
- *v is admissible in G iff $v \subseteq \Gamma_G(v)$,*
- *v is preferred in G iff v is $\subseteq$-maximal admissible in G,*
- *v is complete in G iff $v = \Gamma_G(v)$.*

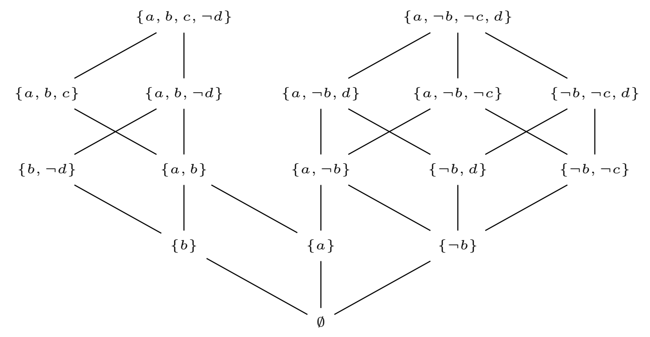
Example 1 *Consider an LAG with $S = \{a, b, c, d\}$ and $L = \{+, -\}$. The following graph shows the labels of each link.*



For simplicity, let's assume all nodes have the same acceptance condition requiring that all positive links must be active (that is the respective parents must be t) and no negative link is active.[5] *We obtain two models, namely $v_1 = \{a, b, c, \neg d\}$ and $v_2 = \{a, \neg b, \neg c, d\}$. The grounded interpretation is $v_3 = \{a\}$. We obtain 16 admissible interpretations:*

[4] The operator in [3] used 3-valued interpretations which are equivalent to partial interpretations. We prefer the latter here as they are conceptually and technically somewhat simpler.

[5] In the pattern language developed in the next section this can be expressed as $\#_t(+) - \#(+) = 0 \wedge \#(-) = 0$.



Among these admissible interpretations $\{a, b, c, \neg d\}$ and $\{a, \neg b, \neg c, d\}$ are preferred. Complete interpretations are these two and in addition $\{a\}$.

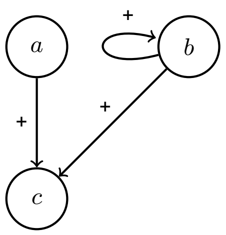
As in logic programming [2], the idea underlying stable semantics here is to exclude self-justifying cycles. Again this semantics can be defined along the lines of the corresponding definition for ADFs in [3]: take a model v, reduce the LAG based on v and check whether the grounded extension of the reduced LAG coincides with the nodes true in v. Here is the definition:

Definition 5 *Let $G = (S, E, L, \lambda, \alpha)$ be an LAG, v a model of G, $S^v = v \cap S$. v is a stable model of G iff v restricted to S^v is the grounded interpretation of $G^v = (S^v, E^v, L, \lambda^v, \alpha^v)$, the v-reduct of G, where*

- *$E^v = E \cap (S^v \times S^v)$,*
- *λ^v is λ restricted to S^v,*[6]
- *α^v is α restricted to S^v.*

Observe that in α^v we did not have to alter the values of the function, i.e. the true and false multisets remain the same (although some of them might become "unused" since the number of parents shrinked). We will see later that this exactly matches the stable semantics for ADFs from [3]. For the moment, we continue our running example.

Example 2 *For Example 1 we obtained two models, $v_1 = \{a, b, c, \neg d\}$ and $v_2 = \{a, \neg b, \neg c, d\}$. In v_1 the justification for b is obviously based on a cycle. The v_1-reduct of our graph is*



It is easy to see that the grounded interpretation of the reduced graph is $\{a\}$, v_1 is thus not a stable model, as intended. We leave it to the reader to verify that v_2 indeed is a stable model.

Results about the semantics carry over from ADFs (see [3]).

Proposition 1 *Let G be an LAG. The following inclusions hold:*

$$stb(G) \subseteq mod(G) \subseteq pref(G) \subseteq com(G) \subseteq adm(G),$$

where $stb(G)$, $mod(G)$, $pref(G)$, $com(G)$ and $adm(G)$ denote the sets of stable models, models, preferred interpretations, complete interpretations and admissible interpretations of G, respectively. Moreover, $pref(G) \neq \emptyset$, whereas $mod(G') = \emptyset$ for some LAG G'.

[6] Given a function $f : M \to N$ and $M' \subseteq M$, f restricted to M' is the function $f' : M' \to N$ such that $f'(m) = f(m)$ for all $m \in M'$.

3 ACCEPTANCE PATTERNS

The definition of the semantics of LAGs introduced in the last section depends on a function assigning one of the truth values t, f to each multiset of labels. In this section we address the question how to represent this acceptance function. We will introduce a specific pattern language for this purpose.

Although in principle there are infinitely many multisets of labels, even if the set of labels is finite, we only need to consider a finite number of multisets, assuming that each node in an LAG has only finitely many parents: the number of occurrences of each label in the relevant multisets is obviously restricted by the number of incoming links with that label. Similar to the way propositional formulas describe Boolean functions by specifying the conditions interpretations have to satisfy to be evaluated to t, we will use a language for specifying conditions a multiset has to satisfy to be evaluated to t. In other words, a pattern will just be a predicate on multisets.

Before we define our pattern language let's look at some examples. Let's start with qualitative labels. Assume $L = \{\text{++}, \text{+}, \text{-}, \text{--}\}$ representing strong support, support, attack and strong attack, respectively and consider the multiset $m_1 = \{\text{++}, \text{+}, \text{-}, \text{-}\}$. Assume further we want to accept a node if its support is stronger than the attack against it, measuring strength, say, by counting support, respectively attack links, multiplying strong support/attack with a factor of 2. m_1 obviously satisfies this acceptance condition as it has 2 active supporting labels, one of them counting twice.

To be able to express conditions like this one we need to be able to refer to the number of occurrences of a label in a multiset. We will use the symbol $\#$ followed by a particular label for this purpose. The condition informally described above can then be represented as

$$2(\#\text{++}) + (\#\text{+}) - 2(\#\text{--}) - (\#\text{-}) > 0.$$

To be able to express conditions like: *half of the positive links must be active* we also make it possible to refer to the total number of links (including those which are not active) with a particular label via the term $\#_t$.

Now consider quantitative labels where L is, say, the set of integers. Consider the multiset $m_2 = \{5, 2, -3, -3\}$. In this case the number of occurrences of a particular number seems less relevant, and we are probably more interested in, say, summing up the values in m_2, or we may want to compare the strongest positive number with the strongest negative one. We will take this into account by providing in our pattern language key words representing the sum, minimum and maximum of the elements in a multiset consisting entirely of numbers (following standard database query languages). Finally, for both qualitative and quantitative labels, we provide a handle for counting different (active or all) labels.

We call LAGs whose acceptance function is defined as a pattern in our pattern language GRAPPA systems (GRaph-based Argument Processing with Patterns of Acceptance).

Definition 6 *A GRAPPA system is a tuple $G = (S, E, L, \lambda, \pi)$ where S, E, L and λ are as in Def. 3 (definition of LAGs) and*

- *$\pi : S \to P^L$ assigns acceptance patterns over L to nodes.*

P^L here denotes the set of acceptance patterns over L defined next:

Definition 7 *Let L be a set of labels.*

- *A term over L is of the form*
  - $(\#l)$, $(\#_t l)$ *for arbitrary $l \in L$,*
  - min, min_t, max, max_t, sum, sum_t, $count$, $count_t$.
- *A basic acceptance pattern (over L) is of the form*

$$a_1 t_1 + \cdots + a_n t_n R\, a$$

where the t_i are terms over L, the a_i's and a are integers and $R \in \{<, \leq, =, \neq, \geq, >\}$.
- *An acceptance pattern (over L) is a basic acceptance pattern or a boolean combination of acceptance patterns.*

We now define the semantics of acceptance patterns in a GRAPPA system G. Whether a multiset of labels satisfies a pattern or not may depend on the node s in the graph where it is evaluated. Terms yield numbers. The value of terms indexed with t is independent of the multiset, it entirely depends on the node s, more precisely on the labels of links with target s. We call these terms node-dependent. In contrast, the non-indexed terms are evaluated based on a given multiset alone.

Definition 8 *Let $G = (S, E, L, \lambda, \pi)$ be a GRAPPA system. For $m : L \to \mathbb{N}$ and $s \in S$ the value function val_s^m is defined as:*[7]

$$\begin{aligned}
val_s^m(\#l) &= m(l)\\
val_s^m(\#_t l) &= |\{(e,s) \in E \mid \lambda((e,s)) = l\}|\\
val_s^m(min) &= \textbf{\textit{min}}\ m\\
val_s^m(min_t) &= \textbf{\textit{min}}\{\lambda((e,s)) \mid (e,s) \in E\}\\
val_s^m(max) &= \textbf{\textit{max}}\ m\\
val_s^m(max_t) &= \textbf{\textit{max}}\{\lambda((e,s)) \mid (e,s) \in E\}\\
val_s^m(sum) &= \textstyle\sum_{l \in L} m(l)\\
val_s^m(sum_t) &= \textstyle\sum_{(e,s) \in E} \lambda((e,s))\\
val_s^m(count) &= |\{l \mid m(l) > 0\}|\\
val_s^m(count_t) &= |\{\lambda((e,s)) \mid (e,s) \in E\}|
\end{aligned}$$

The satisfaction relation $\models$ for basic patterns is given by:

$$(m,s) \models a_1 t_1 + \cdots + a_n t_n R\, a \quad \textit{iff} \quad \sum_{i=1}^{n} \Big(a_i\ val_s^m(t_i)\Big)\ R\ a.$$

Satisfaction of boolean combinations is inductively defined as usual, e.g. for acceptance patterns p_1 and p_2 we have $(m,s) \models p_1 \wedge p_2$ iff $(m,s) \models p_1$ and $(m,s) \models p_2$, $(m,s) \models \neg p_1$ iff $(m,s) \not\models p_1$, etc.

This puts us in a position to establish the connection with LAGs. For each node s with pattern $\pi(s)$, the function $\alpha(s)$ associated with $\pi(s)$ is defined as

$$\alpha(s)(m) = t \text{ iff } (m,s) \models \pi(s).$$

GRAPPA systems thus are LAGs whose acceptance functions are defined by patterns. Note that we can now define the characteristic operator for a GRAPPA system $G = (S, E, L, \lambda, \pi)$ equivalently as $\Gamma_G(v) = P_G(v) \cup N_G(v)$ with

$$P_G(v) = \{s \mid (m,s) \models \pi(s) \text{ for each } m \in \{m_s^{v'} \mid v' \in [v]_c\}\},$$
$$N_G(v) = \{\neg s \mid (m,s) \not\models \pi(s) \text{ for each } m \in \{m_s^{v'} \mid v' \in [v]_c\}\}.$$

[7] ***min*** m and ***max*** m return the minimal, respectively maximal element of a multiset m and are undefined in case $m = \emptyset$ or m contains a non-numerical term. The sum over labels is undefined whenever one of the summands is non-numerical, it equals 0 whenever there are no summands.

For convenience we introduce some useful abbreviations:

- in cases where readability is not affected we omit brackets;
- whenever $a_i = -b_i$ is a negative number we write $-b_i t_i$ rather than $+a_i t_i$ in basic acceptance patterns;
- we use $\#\{l_1, \ldots, l_n\}$ for $\#l_1 + \ldots + \#l_n$, the same for $\#_t l$;
- we also use conditional acceptance patterns of the form $\phi_1 : \phi_2/\phi_3$, where ϕ_i are acceptance patterns, as an abbreviation for $\phi_1 \wedge \phi_2 \vee \neg\phi_1 \wedge \phi_3$.

Lemma 1 *Let $L = \{l_1, \ldots, l_k\}$ be a finite set of labels. Each finitary L-acceptance function (the total number of occurrences of labels in each multiset evaluated to t is finite) can be represented as an acceptance pattern over L.*

Proof Let g be a finitary L-acceptance function. Each finite multiset m evaluated to t can be represented as a conjunction $\#l_1 = m(l_1) \wedge \ldots \wedge \#l_k = m(l_k)$. The pattern for g then is the disjunction of all conjunctions representing multisets evaluated to t.

As we have seen, the semantics of some patterns depends on the graph they are used in. This raises an important issue related to stable semantics where stability is checked via a reduction of the original graph (see Def. 5). Note that the semantics of node-independent terms is not affected by the reduction as these terms actually depend on the labels of active links which are preserved. On the other hand, the meaning of node-dependent terms (those indexed by t) may change in the reduced graph. This has an important consequence: we have to replace node-dependent terms in patterns by their actual values in the original graph before using them in the reduced graph. It is easy to see that this can always be done. For instance, for node c in the graph of Example 1 the original acceptance pattern $\#_t(+) - \#(+) = 0 \wedge \#(-) = 0$ becomes $2 - \#(+) = 0 \wedge \#(-) = 0$ which is equivalent to the pattern $\#(+) = 2 \wedge \#(-) = 0$.

4 USE CASES

In this section we illustrate how some typical argument graphs can be reconstructed using GRAPPA.

Bipolar argument graphs and Dung frameworks

In Sect. 1 we used bipolar argument graphs with labels for support (+) and attack (-) as a motivating example. The acceptance conditions discussed there are expressed as follows:

- all positive, no negative link active: $(\#_t+) - (\#+) = 0 \wedge (\#-) = 0$,
- at least one positive, no negative active link: $(\#+) > 0 \wedge (\#-) = 0$,
- more positive than negative active links: $(\#+) - (\#-) > 0$.

For an alternative treatment of bipolar frameworks, see [1, 6]. Dung frameworks have no labels, yet they can be viewed as having the single label - left implicit. They use a single pattern for all nodes:

- no negative active link: $(\#-) = 0$

Let's call this pattern the Dung pattern. We have the following result:

Proposition 2 *Let $F = (A, R)$ be a Dung framework. The associated GRAPPA system is $G_F = (A, R, \{-\}, \lambda, \alpha)$ where α assigns the Dung pattern to all nodes. E is grounded, admissible, complete, preferred, stable wrt. F iff $E = v \cap A$ for some grounded, admissible, complete, preferred, stable interpretation v of G_F.*

Weighted argument graphs

Weighted graphs have as labels positive or negative numbers, expressing the strength of support, respectively attack (see also [7]). Again various patterns come into mind:

- the sum of weights of active links is greater than 0: $sum > 0$.
- the highest active support is stronger than the strongest (lowest) attack: $max + min > 0$
- the difference among strongest active support and the strongest active attack is above some threshold b: $max + min > b$.

Farley/Freeman proof standards

Farley and Freeman [11] introduced a framework for expressing 5 different proof standards based on 4 different types of arguments: valid, strong, credible and weak arguments. The strength of the argument types is decreasing in the given order. Since arguments can be pro or con a particular proposition, we need 8 labels $v, s, c, w, -v, -s, -c, -w$. The $-$ expresses a con argument of the respective type. The proof standards discussed by Farley and Freeman can be captured using the following patterns:

- scintilla of evidence: $\#\{v, s, c, w\} > 0$
- preponderance of evidence:
  $\#\{v, s, c, w\} > 0 \wedge (\# -v) = 0 \wedge (\# -s) = 0 \vee$
  $(\#v) > 0 \wedge (\# -c) = 0 \vee$
  $\#\{v, s\} > 0 \wedge (\# -w) = 0 \vee$
  $\#\{v, s, c\} > 0$
- dialectical validity: $\#\{v, s, c\} > 0$, $\#\{-v, -s, -c, -w\} = 0$
- beyond reasonable doubt: $\#\{v, s\} > 0$, $\#\{-v, -s, -c, -w\} = 0$
- beyond doubt: $\#v > 0$, $\#\{-v, -s, -c, -w\} = 0$

Rather than assigning a particular proof standard to each node, it may be useful in some (legal) settings to have dynamic proof standards, that is to make it possible to argue about the proof standard in the same way as about any other topic. Assume it is an issue whether the proof standard for a node s is, say, dialectical validity (dv) or beyond reasonable doubt (brd). Let's assume dv is the default, and the stronger proof standard is only applied if this was established during the argumentation, that is if there is some parent node p of s representing the information that brd is needed. We introduce an additional label brd and assign it to the link (p, s). The new, dynamic proof standard can conveniently be represented using the conditional pattern

$$\#brd > 0 : patt(brd)/patt(dv).$$

Here $patt(ps)$ is the pattern for proof standard $ps \in \{brd, dv\}$.

ADFs

Acceptance conditions of ADFs are propositional formulas built from parent nodes rather than labels of links. To model ADFs in GRAPPA we just have to label each link with its source node, that is, the set of labels L is identical to the set of nodes S and for each link $l = (p, s)$ we have $\lambda(l) = p$. The acceptance pattern for each node s is obtained from its ADF acceptance condition C_s by simply replacing each occurrence of an atom a in C_s by the basic pattern $\#a = 1$.

Proposition 3 *Let A be an ADF, G_A the GRAPPA system obtained from A as described above. A and G_A are equivalent under all semantics introduced in Sect. 2.*

Carneades

Carneades [13, 14] is an advanced model of argumentation capturing, among other things, weighted arguments and 5 different proof standards. In [4] Carneades was reconstructed using ADFs and generalized to arbitrary cyclic graph structures. It is thus apparent that Carneades can be modelled in GRAPPA. Nevertheless, we give a direct reconstruction here as it is simpler than the one obtained indirectly via ADFs.

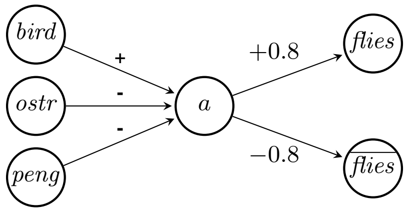
A Carneades argument is a tuple $\langle P, E, c\rangle$ with premises P, exceptions E ($P \cap E = \emptyset$) and conclusion c. c and elements of P, E are literals. An argument evaluation structure (CAES) is a tuple $C = \langle args, ass, weight, standard\rangle$, where

- $args$ is a set of arguments,
- ass is a consistent set of literals, the assumptions,
- $weight$ assigns a real number to each argument, and
- $standard$ maps propositions to a proof standard.

Assumptions are best handled as modifiers of the original arguments: $args^{ass}$ is obtained from $args$ by

1. deleting each argument $\langle P, E, c\rangle$ where $\overline{q} \in ass$ for some $q \in P$ or $q \in ass$ for some $q \in E$,[8] and
2. deleting each $q \in ass$ from the premises of the remaining arguments, and
3. deleting each q such that $\overline{q} \in ass$ from the exceptions of the remaining arguments.

The graph G^{ass}_{args} is then obtained as in [4] by translating each argument. We illustrate the translation using an example: $a = \langle\{bird\}, \{peng, ostr\}, flies\rangle$ with $weight(a) = 0.8$ translates to:



The graph obtained this way uses two types of nodes, proposition and argument nodes. Labels for links to argument nodes are + and -, labels for links to proposition nodes positive/negative reals. The pattern for argument nodes is

- $(\#_t+) - (\#+) = 0 \wedge (\#-) = 0$,

Patterns for proposition nodes depend on the proof standards (α, β and γ are positive numerical parameters):

- scintilla of evidence: $max > 0$
- preponderance of evidence: $max + min > 0$
- clear and convincing evidence: $max > \alpha \wedge max + min > \beta$
- beyond reasonable doubt: $max > \alpha \wedge max + min > \beta \wedge -min < \gamma$
- dialectical validity: $max > 0 \wedge min > 0$

Proposition 4 *For each CAES C the GRAPPA system obtained by the construction described above is equivalent to the ADF as obtained in [4] under all semantics.*

5 COMPUTATIONAL ASPECTS

In this section we show how LAGs and GRAPPA systems can be translated to equivalent ADFs. An ADF is a tuple (S, E, C) where S is a set of statements; $E \subseteq S \times S$ is a set of links; and $C = \{C_s\}_{s \in S}$ is a set of propositional formulas such that C_s is given over atoms $par_E(s) = \{t \in S \mid (t, s) \in E\}$.

For the forthcoming definition, recall the notion of m^v_s from Definition 3. Given an LAG G, let $s \in S$ and $T \subseteq par_E(s)$. The multiset of active labels of s based on T, m^T_s, is the unique multiset defined as: $m^T_s = m^v_s$ for some partial interpretation v assigning t to each node in T and f to each node in $par_E(s) \setminus T$.

[8] $\overline{q}$ is the complement of literal q.

Definition 9 *For an LAG $G = (S, E, L, \lambda, \alpha)$ define its associated ADF A_G as (S, E, C_G) where*

$$C_G(s) = \bigvee_{T \subseteq par_E(s) : \alpha(s)(m_s^T) = t} \left(\bigwedge_{r \in T} r \wedge \bigwedge_{r \in par_E(s) \setminus T} \neg r \right).$$

Proposition 5 *Let G be an LAG, A_G its associated ADF constructed as described above. G and A_G are equivalent under all semantics introduced in Sect. 2.*

Proof Sketch: we can show that the characteristic operators of both G and A_G are equivalent. From this the result follows for all semantics except stable. For stable, we can additionally show that reducts are equivalent, i.e. for each model v of G, A_{G^v} is the same object as the ADF A_G reduced w.r.t. v (see [3], Def.6).

For a GRAPPA system $G = (S, E, L, \lambda, \pi)$ where acceptance functions are represented as patterns we can express the ADF acceptance conditions accordingly:

$$C_G(s) = \bigvee_{T \subseteq par_E(s) : (m_s^v, s) \models \pi(s)} \left(\bigwedge_{r \in T} r \wedge \bigwedge_{r \in par_E(s) \setminus T} \neg r \right).$$

This reduction paves the way for implementations via the ADF translations and the use of the existing ADF system Diamond [10].

Finally, we briefly address the complexity of reasoning with GRAPPA systems. All hardness results for ADFs (see [17] for a detailed overview) carry over to GRAPPA systems thanks to the poly-time translation of ADFs to GRAPPA systems given in the previous section. Concerning membership results note that they do *not* follow directly from the reductions given above, since those might have an exponential blow-up. However, it is rather easy to see that complexity does not increase compared to ADFs since evaluating the characteristic operator for a GRAPPA system is equally hard as evaluating the characteristic operator for ADFs. To be more precise, consider the following problem: given a GRAPPA system $G = (S, E, L, \lambda, \pi)$, a node $s \in S$, and a partial interpretation v, decide whether $s \in \Gamma_G(v)$, respectively $\neg s \in \Gamma_G(v)$. The problem is shown to be in coNP by the following complementary nondeterministic algorithm: Guess $v' \in [v]_c$ and check $(m_s^{v'}, s) \not\models \pi(s)$ (resp. $(m_s^{v'}, s) \models \pi(s)$). These checks can be done in polynomial time, since evaluating a basic acceptance pattern does not involve more than counting, finding minimal or maximal elements, and some simple arithmetics; also computing the outcome for a Boolean combination of basic acceptance patterns is then straightforward. With this result the verification problem for admissible semantics is easily seen to be in coNP as well (matching the coNP result for ADFs for this problem). Further membership results for other semantics then follow in the same way as discussed in [17]. Thus, GRAPPA systems provide a convenient and powerful language for specifying acceptance functions without additional costs as compared to ADFs.

6 DISCUSSION

In this paper we introduced a semantical framework that allows us to define Dung-style semantics for arbitrary labelled graphs. The approach rests on acceptance functions based on multisets of labels. We introduced a pattern language for representing such functions, gave various examples demonstrating the expressiveness of the approach, and showed how it can be implemented via a translation to ADFs.

Dov Gabbay [12] analyzed argument (and other) graphs from an equational point of view. This interesting work is very general and highly abstract. However, it is far from immediate how it could directly be applied to the goals of this paper. In particular, our operator-based semantic definitions for LAGs do not have a correspondence in Gabbay's work.

Analyzing various attempts to generalize Dung frameworks, Modgil [15] recently coined the term abstract locution frameworks, that is frameworks which represent the way people express their views in their communication. We believe GRAPPA can be very helpful for the specification of such frameworks, and in particular for equipping them with a precise formal semantics, the necessary prerequisite for automatic evaluation.

As to future work, we want to investigate refinements of the ADF-based implementation: to keep acceptance conditions of the resulting ADF simple, techniques from SAT-based constraint solving [16] could replace the naive translation given above. Another option is to extend the Diamond system directly using special features of ASP systems like weight constraints and aggregates. We also plan to explore LAGs where (i) *sets* of labels are assigned to edges; and (ii) labels come with an internal structure, e.g. *preferences* among them.

REFERENCES

[1] Leila Amgoud, Claudette Cayrol, Marie-Christine Lagasquie, and Pierre Livet, 'On Bipolarity in Argumentation Frameworks', *International Journal of Intelligent Systems*, **23**, 1–32, (2008).

[2] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński, 'Answer set programming at a glance', *Commun. ACM*, **54**(12), 92–103, (2011).

[3] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran, 'Abstract dialectical frameworks revisited', in *Proc. IJCAI*, pp. 803–809. IJCAI/AAAI, (2013).

[4] Gerhard Brewka and Thomas F. Gordon, 'Carneades and abstract dialectical frameworks: A reconstruction', in *Proc. COMMA*, pp. 3–12. IOS Press, (2010).

[5] Gerhard Brewka and Stefan Woltran, 'Abstract dialectical frameworks', in *Proc. KR*, pp. 102–111. AAAI Press, (2010).

[6] Claudette Cayrol and Marie-Christine Lagasquie-Schiex, 'Bipolarity in argumentation graphs: Towards a better understanding', *Int. J. Approx. Reasoning*, **54**(7), 876–899, (2013).

[7] Sylvie Coste-Marquis, Sébastien Konieczny, Pierre Marquis, and Mohand Akli Ouali, 'Weighted attacks in argumentation frameworks', in *Proc. KR*, eds., Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, pp. 593–597. AAAI Press, (2012).

[8] Marc Denecker, Victor W. Marek, and Mirosław Truszczyński, 'Ultimate approximation and its application in nonmonotonic knowledge representation systems', *Inf. Comput.*, **192**(1), 84–121, (2004).

[9] Phan Minh Dung, 'On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games', *Artif. Intell.*, **77**, 321–357, (1995).

[10] Stefan Ellmauthaler and Hannes Strass, 'The DIAMOND system for argumentation: Preliminary report', in *Proc. ASPOCP*, eds., Michael Fink and Yuliya Lierler, (2013).

[11] Arthur M. Farley and Kathleen Freeman, 'Burden of proof in legal argumentation', in *Proc. ICAIL'95*, pp. 156–164. IOS Press, (1995).

[12] Dov M. Gabbay, 'Equational approach to argumentation networks', *Argument & Computation*, **3**(2-3), 87–142, (2012).

[13] Thomas F. Gordon, Henry Prakken, and Douglas Walton, 'The Carneades model of argument and burden of proof', *Artif. Intell.*, **171**(10-15), 875–896, (2007).

[14] Thomas F. Gordon and Douglas Walton, 'Proof burdens and standards', in *Argumentation in Artificial Intelligence*, eds., Iyad Rahwan and Guillermo Simari, 239–258, Springer, (2009).

[15] Sanjay Modgil, 'Revisiting abstract argumentation', in *Proc. TAFA*, eds., Liz Black, Sanjay Modgil, and Nir Oren, (2013).

[16] Takehide Soh, Naoyuki Tamura, and Mutsunori Banbara, 'Scarab: A rapid prototyping tool for SAT-based constraint programming systems', in *Proc. SAT*, eds., Matti Järvisalo and Allen Van Gelder, volume 7962 of *LNCS*, pp. 429–436. Springer, (2013).

[17] Hannes Strass and Johannes Peter Wallner, 'Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory', in *Proc. KR, to appear*. AAAI Press, (2014).