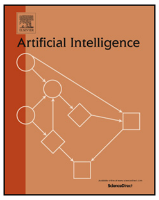
Artificial Intelligence 233 (2016) 1–59

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



Argument graphs and assumption-based argumentation [☆]

Robert Craven [*], Francesca Toni

Department of Computing, 180 Queen's Gate, Imperial College London, London, SW7 2AZ, United Kingdom

ARTICLE INFO

Article history:
Received 1 April 2015
Received in revised form 11 December 2015
Accepted 16 December 2015
Available online 23 December 2015

Keywords:
Argumentation

ABSTRACT

Arguments in structured argumentation are usually defined as trees, and extensions as sets of such tree-based arguments with various properties depending on the particular argumentation semantics. However, these arguments and extensions may have redundancies as well as circularities, which are conceptually and computationally undesirable. Focusing on the specific case of Assumption-Based Argumentation (ABA), we propose novel notions of arguments and admissible/grounded extensions, both defined in terms of graphs. We show that this avoids the redundancies and circularities of standard accounts, and set out the relationship to standard tree-based arguments and admissible/grounded extensions (as sets of arguments). We also define new notions of graph-based admissible/grounded dispute derivations for ABA, for determining whether specific sentences hold under the admissible/grounded semantics. We show that these new derivations are superior with respect to standard dispute derivations in that they are complete in general, rather than solely for restricted classes of ABA frameworks. Finally, we present several experiments comparing the implementation of graph-based admissible/grounded dispute derivations with implementations of standard dispute derivations, suggesting that the graph-based approach is computationally advantageous.

1. Introduction

Argumentation theory is a powerful reasoning abstraction in which conflicting arguments are represented and evaluated against one another in order to resolve conflicts and find those sets of arguments which are together dialectically superior. It has been extensively studied in AI over the past two decades—see [2,4,24] for an overview—and used as the formal basis of a number of applications. Several forms of argumentation have been proposed. The simplest form is the seminal *abstract argumentation* defined by Dung [11], where the basic structure is a graph whose vertices represent *arguments* and whose edges, called *attacks*, represent a relation of conflict between arguments. By contrast, in *structured argumentation*—see [3] for an overview—arguments and attacks are not primitive but are derived from more basic structures. It is common in structured argumentation to define arguments as trees, whose edges represent a relation of dependency holding between sentences labelling the nodes.

Assumption-Based Argumentation (ABA) [6,12,14,13,27,28] is a well-known form of structured argumentation. In ABA, arguments are obtained from the *rules* of a given deductive system and *assumptions* (special sentences in the language

[☆] The work presented herein was supported by EPSRC grant EP/J020915/1 (TRaDAr—Transparent Rational Decisions by Argumentation). All data and programs used are available from http://robertcraven.org/proarg/.
[*] Corresponding author.
E-mail addresses: robert.craven@gmail.com (R. Craven), ft@doc.ic.ac.uk (F. Toni).

http://dx.doi.org/10.1016/j.artint.2015.12.004
0004-3702/© 2015 Published by Elsevier B.V.

underlying the deductive system). More specifically, arguments are finite trees whose leaves must either be labelled by assumptions or must represent the empty body of a rule in the deductive system. Such an argument is attacked by another argument when an assumption on which the first argument is built has as *contrary* a sentence labelling the root of the second argument. In ABA, the internal structure of arguments is explicit, as is also the reason why there is an attack between two arguments.

The semantics of argumentation frameworks typically determine different dialectically superior or winning sets of arguments, known as *acceptable extensions*. Both abstract argumentation and ABA define various alternative semantics and corresponding kinds of acceptable extensions. In the case of ABA, extensions can be equivalently understood in terms of sets of assumptions (in the support of arguments in acceptable extensions)—see [6,14,28].

ABA has been applied in several settings, e.g., to support medical decision-making [8,19] and e-procurement [22]. ABA's applicability relies on the existence of computational mechanisms, based on various kinds of *dispute derivation* [12,14,27] that are formally proven to be correct procedures under various semantics. Whereas the semantics are non-constructive specifications of what can be deemed acceptable extensions, dispute derivations are fully constructive algorithms. One kind of dispute derivation for computation under the semantics of admissible extensions was presented by Dung et al. [12]; this was extended to the semantics of grounded and ideal extensions by Dung et al. [14], and, in a parametric version with a richer output (encompassing both views of extensions as sets of arguments and as sets of assumptions) by Toni [27].

ABA is not alone among forms of structured argumentation in representing arguments as tree structures; [23] and others do the same. This has several consequences. Positively, it means that the relation of support is depicted explicitly in the formalism, with an edge of such a tree-based argument representing the relation of dependence of one sentence on others. Yet, negatively, it can lead to several problems, both conceptual and computational. First, defining arguments as trees whose nodes are labelled by sentences means that there can be circular dependencies amongst those sentences, even if these trees are required to be finite. The potential for circular dependency also causes problems computationally: it means that, in the course of a dispute derivation, loops may be encountered which prevent standard procedures from terminating, leading to incompleteness. Secondly, even if there is no circular dependency, the use of trees to represent arguments allows a sentence to be proved in several different ways (which we call *flabbiness*). Flabbiness is conceptually undesirable and involves redundancy, with the same sentence being reproved needlessly; it is also therefore inefficient.

Thirdly, some of these issues arise not only with the definition and computation of individual arguments, but also with the definition and computation of extension-based semantics, i.e., with sets of arguments. These sets are intended to represent a coherent dialectical position, but, as we discuss in the paper, if the same sentence is proved in multiple different ways in different arguments belonging to the set (which we call *bloatedness*), it can rightly be questioned whether an appropriate notion of coherence applies. Indeed, one may have already computed that there is an argument for some sentence, but not use this where it could serve as a subargument elsewhere. Again, as with the case of the flabbiness of individual arguments, there is also a question of efficiency in the computation of extensions.

In the current paper we provide a solution, which answers the conceptual problems, as well as the computational issues of incompleteness and inefficiency. The solution relies upon altering the underlying conception of an argument to an approach which is graph-based rather than tree-based, and which also reconceives the nature of the set of arguments sanctioned by the semantics and computed in a dispute derivation, removing redundancy across arguments. Using graphs rather than sets of trees, circularity, flabbiness and bloatedness are removed at a stroke. Our work focuses on the case of ABA in particular, but we think that the approach of using graphs would also generalize to other forms of structured argumentation which are currently based on trees, such as [23].

We provide a link between our argument-graph approach and *rule-minimal* arguments, which makes the connection to standard, tree-based accounts of arguments in ABA explicit. We define novel *admissible* and *grounded* graph-based semantics for our argument graphs, and show the correspondence with the standard semantics using trees. We then define dispute derivations for the new structures, which we prove are sound and complete with respect to the novel semantics. Completeness in the dispute derivations for grounded semantics is an important further advantage of our approach, for previous dispute derivations for ABA—those of Dung et al. [12], Dung et al. [14] and Toni [27]—are complete solely for a special form of (p-acyclic) ABA frameworks. Indeed, our dispute derivations are complete for any form of ABA framework.

In addition to the gains in conceptual justification and completeness, there are improvements in the speed with which computation is performed in the new approach. We implemented our algorithms in Prolog,[1] and conducted a preliminary experimental evaluation of the new algorithms in comparison with an implementation of the previous dispute derivations of Toni [27]. The results, as can be seen in §6, favour the graph-based approach.[2]

The work here substantially extends the preliminary research in [9]. First, the previous work was restricted to the grounded semantics, still defined in terms of sets of tree-based arguments rather than graphs. Secondly, the main results of that paper were given as proof sketches; full proofs are now provided. Thirdly, the previous paper focused on soundness; full completeness results are now also provided. Fourthly, we conduct a more thorough experimental comparison with standard dispute derivations. Fifthly, many more examples are provided. Sixthly, in previous work argument graphs were conceived more as a data structure to aid computation; in the present paper they are justified on more conceptual grounds,

[1] Implementations and several ABA frameworks freely available at http://robertcraven.org/proarg/.

[2] Experimental data and results available from http://robertcraven.org/proarg/experiments.html.

as an appropriate representation of the justification structure in arguments. Seventhly, and related to the previous point, we provide a new semantics for argument graphs.

The paper is organized as follows. In §2 we give background on ABA. In §3 we describe conceptual and associated computational problems with the existing formulation. In §4 we describe our formalism and relate it to existing definitions and semantics for ABA. In §5 we give dispute derivations for argument graphs, proving soundness and completeness with respect to admissible and grounded semantics. In §6 we discuss experiments comparing standard dispute derivations with the graph-based derivations introduced in the previous section. In §7 we compare our contribution in this paper to related work. In §8 we conclude and discuss future research. Proofs not included in the main text are in the Appendix.

2. Background

An *ABA framework* [6,13,28] is a tuple $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$:

- $(\mathcal{L}, \mathcal{R})$ is a deductive system, with $\mathcal{L}$ a set of *sentences* and $\mathcal{R}$ a set of *(inference) rules* of the form $s_0 \leftarrow s_1, \ldots, s_m$, for $m \geqslant 0$ and $s_0, s_1, \ldots, s_m \in \mathcal{L}$;
- $\mathcal{A} \subseteq \mathcal{L}$ is a non-empty set, the *assumptions*;
- $\overline{}$ is a total mapping from $\mathcal{A}$ to $\mathcal{L}$, with $\bar{a}$ known as the *contrary* of a.

A *flat* ABA framework is an ABA framework such that for no rule $s_0 \leftarrow s_1, \ldots, s_m \in \mathcal{R}$ does it hold that $s_0 \in \mathcal{A}$. In the remainder of the paper, unless otherwise specified, we assume as given a flat ABA framework $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$.

In ABA, arguments are proofs using rules in $\mathcal{R}$, with each argument ultimately depending on assumptions [13,28]. Proofs and arguments are standardly defined as trees:

- A *proof for* $s \in \mathcal{L}$ *supported by* $S \subseteq \mathcal{L}$ is a (finite) tree with nodes labelled by sentences in $\mathcal{L}$ or $\top \notin \mathcal{L}$, where the root is labelled by s and:
  - all non-leaf nodes n are labelled by some $s_0 \in \mathcal{L}$, such that there is some rule $s_0 \leftarrow s_1, \ldots, s_m \in \mathcal{R}$ with either (i) $m = 0$ and the unique child of n is labelled by $\top$, or (ii) $m > 0$ and n has m children, labelled by $s_1, \ldots, s_m$ respectively; and
  - S is the set of all sentences in $\mathcal{L}$ labelling the leaves.[3]
- An *argument for* $s \in \mathcal{L}$ is a proof for s supported by some $A \subseteq \mathcal{A}$.[4] (We sometimes call such arguments *tree-based arguments*, in order to distinguish them from the argument graphs we later introduce.)

We will use the following notation regarding the structure of arguments. Where a is an argument for s supported by A, $claim(\mathsf{a}) = s$ (s is the *claim* of a) and $support(\mathsf{a}) = A$ (A is the *support* of a). Where $A \subseteq \mathcal{A}$, then $args(A)$ is the set $\{\mathsf{a} \mid support(\mathsf{a}) \subseteq A\}$, i.e., the set of arguments whose support is a subset of A. Where A is a set of arguments (an *extension*), $claims(\mathsf{A})$ is $\{claim(\mathsf{a}) \mid \mathsf{a} \in \mathsf{A}\}$. Where a is an argument, we let $nodes(\mathsf{a})$ be the set of nodes of a. Where $n \in nodes(\mathsf{a})$, $label(n)$ is the sentence, or $\top$, which labels n and $children(n, \mathsf{a})$ is the (possibly empty) set of children of n in a; where a is clear from the context, we write $children(n, \mathsf{a})$ simply as $children(n)$; where N is a set of nodes of a proof, $labels(N)$ is $\{label(n) \mid n \in N\}$.

In ABA the attack relation between arguments is defined in terms of assumptions and their contraries:

- an argument a *attacks* an argument b (written here as $\mathsf{a} \rightsquigarrow \mathsf{b}$) iff there is some $b \in support(\mathsf{b})$ such that $\bar{b} = claim(\mathsf{a})$.

This notion is lifted to sets as follows: a set of arguments A attacks a set of arguments B (written here as $\mathsf{A} \rightsquigarrow \mathsf{B}$) iff some $\mathsf{a} \in \mathsf{A}$ attacks some $\mathsf{b} \in \mathsf{B}$; an argument a attacks a set of arguments B (written here $\mathsf{a} \rightsquigarrow \mathsf{B}$) iff $\mathsf{a} \rightsquigarrow \mathsf{b}$ for some $\mathsf{b} \in \mathsf{B}$; and a set of arguments A attacks an argument b (written here $\mathsf{A} \rightsquigarrow \mathsf{b}$) iff some $\mathsf{a} \in \mathsf{A}$ attacks b.

Example 1. Consider the ABA framework $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$ where:

$$\begin{aligned} \mathcal{L} &= \{\, p, q, r, s, a, b \,\} \\ \mathcal{R} &= \{\, p \leftarrow q, r, \\ &\qquad q \leftarrow, \\ &\qquad r \leftarrow a, \\ &\qquad s \leftarrow b \,\} \end{aligned}$$

[3] Note that $\top$ may label a leaf, $\top$ is not included in S since $\top \notin \mathcal{L}$.

[4] Although these definitions of proof and argument allow for multiple isomorphic proofs and arguments, labelled in the same way, we will ignore this complication in the rest of the paper, and presume that the definitions uniquely specify proofs and arguments; this is common in discussions of ABA and structured argumentation.

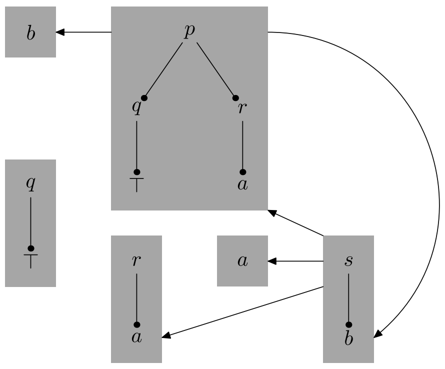


Fig. 1. Arguments and attack relation for Example 1.

$$\mathcal{A} = \{ a, b \}$$
$$\bar{a} = s,\ \bar{b} = p$$

The arguments and attack relation depicted in Fig. 1 are obtained. Here (as throughout the paper), arguments are in shaded boxes, The root of an argument is always at the top of a box and where a sentence within an argument depends on others—say, p on q and r, in the diagram, through the rule $p \leftarrow q, r$—then this is represented by the others' being below the sentence in question and connected to it by a directed edge (the dotted end of the edge is the sentence in the body). Attacks between arguments are shown as arrows. If $\mathsf{a} \rightsquigarrow \mathsf{b}$, we show the arrow as stemming from the vicinity of $claim(\mathsf{a})$ and finishing in the vicinity of some $b \in support(\mathsf{b})$ such that $\bar{b} = claim(\mathsf{a})$. ⌟

Attacks represent conflicts, and argumentation semantics constitute recipes to determine how to resolve these conflicts and determine acceptable (or winning) sets of arguments [11,6,14]. Several alternative notions of acceptable sets of arguments (referred to as *extensions*) have been proposed for ABA [6,14]. Here, we focus on admissible and grounded extensions. An extension is defined to be:

- *conflict-free* iff it does not attack itself;
- *admissible* iff it is conflict-free and attacks every argument attacking it;
- *complete* iff it is admissible and contains all arguments it can *defend* (by attacking all arguments attacking them);
- *grounded* iff it is minimally (w.r.t. $\subseteq$) complete.

A *sentence* $s \in \mathcal{L}$ is *admissible/grounded* (optionally, *w.r.t.* $A \subseteq \mathcal{A}$) iff

- there is an argument a with $claim(\mathsf{a}) = s$ such that $\mathsf{a} \in \mathsf{A}$ for some admissible/grounded extension A (optionally, with $A = \bigcup_{\mathsf{a} \in \mathsf{A}} support(\mathsf{a})$).

Example 1 *(Continued).* Let p^* be the argument for p in Fig. 1, a^* the argument for a in Fig. 1, and so on. (In the present example, these are unambiguously defined.) Then, the admissible extensions are:

$\{\mathsf{p}^*\}$	$\emptyset$
$\{\mathsf{p}^*, \mathsf{q}^*\}$	$\{\mathsf{s}^*\}$
$\{\mathsf{p}^*, \mathsf{r}^*\}$	$\{\mathsf{s}^*, \mathsf{b}^*\}$
$\{\mathsf{p}^*, \mathsf{a}^*\}$	$\{\mathsf{s}^*, \mathsf{q}^*\}$
$\{\mathsf{p}^*, \mathsf{q}^*, \mathsf{r}^*\}$	$\{\mathsf{s}^*, \mathsf{b}^*, \mathsf{q}^*\}$
$\{\mathsf{p}^*, \mathsf{q}^*, \mathsf{a}^*\}$	$\{\mathsf{q}^*\}$
$\{\mathsf{p}^*, \mathsf{r}^*, \mathsf{a}^*\}$	
$\{\mathsf{p}^*, \mathsf{q}^*, \mathsf{r}^*, \mathsf{a}^*\}$	

The complete extensions are just $\{\mathsf{p}^*, \mathsf{q}^*, \mathsf{r}^*, \mathsf{a}^*\}$, $\{\mathsf{s}^*, \mathsf{b}^*, \mathsf{q}^*\}$ and $\{\mathsf{q}^*\}$. The grounded extension—unique, as always—is $\{\mathsf{q}^*\}$. ⌟

Several algorithms for determining acceptability of sentences in ABA have been proposed [12,14,27], starting from the dispute derivations presented by Dung et al. [12], to the generic form, taking parameters which give specific instances for the two mentioned semantics (as well as a third semantics not considered here), presented by Toni [27]. We leave the details as references for the interested reader, but note that the dispute derivations we present in §5 are based on this existing work.

Given a set *Args* of arguments, and an attack relationship $\rightsquigarrow \subseteq (Args \times Args)$, Dung [11] also defines the *characteristic function of the abstract argumentation framework* $(Args, \rightsquigarrow)$ as the function $f : 2^{Args} \to 2^{Args}$ such that, for all $\mathsf{A} \subseteq Args$:

$$f(\mathsf{A}) = \{\mathsf{a} \in Args \mid \forall \mathsf{b} \in Args((\mathsf{b} \rightsquigarrow \mathsf{a}) \to (\mathsf{A} \rightsquigarrow \mathsf{b}))\}.$$

Thus $f(\mathsf{A})$ is the set of all arguments defended by A. The characteristic function provides an alternative means of specifying the various semantics defined above, as well as alternative semantics treated in the literature. Given a conflict-free extension $\mathsf{A} \subseteq Args$, it was shown by Dung [11] that:

- A is admissible iff $\mathsf{A} \subseteq f(\mathsf{A})$;
- A is complete iff $\mathsf{A} = f(\mathsf{A})$;
- A is grounded iff A is the least fixed point of f.

This alternative characterization of the semantics will be important when we give a semantics for our argument graphs, in §4.3.

3. Motivation

In at least those forms of structured argumentation which use trees, or analogues of them, to represent arguments—e.g., ABA and [23]—the conception of an argument enforces a form of relevance of the support to the claim. However, the practice of defining arguments as trees allows undesirable patterns of circularity and redundancy in arguments, even if these trees are required to be finite, as is the case for ABA—see §2. In the current section we investigate and define these forms of redundancy. We focus on ABA specifically, but much of what is said here, and the particular definitions, could easily be adapted to other forms of tree-based structured argumentation.

As a basis for discussion, consider the following example in ABA.

Example 2. Consider the ABA framework $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$ where:

$$\begin{aligned}
\mathcal{L} &= \{\, p, q, r, s, x, a, b \,\} \\
\mathcal{R} &= \{\, p \leftarrow q, r, \\
&\qquad p \leftarrow b, \\
&\qquad q \leftarrow p, \\
&\qquad q \leftarrow r, \\
&\qquad r \leftarrow a, \\
&\qquad r \leftarrow b, \\
&\qquad s \leftarrow r \,\} \\
\mathcal{A} &= \{\, a, b \,\} \\
\bar{a} &= x, \ \bar{b} = x
\end{aligned}$$

Fig. 2 shows four trees: a_2–a_4 are arguments for p, and a_1 is an infinite tree which does not qualify as an argument. Fig. 3 shows two extensions. Both extensions are admissible. However, A contains one argument in which r is supported by a and another in which r is supported by b. ⌟

In Example 2, each sentence in the infinite tree a_1 depends on other sentences, as determined by the rules in $\mathcal{R}$; in that sense, the tree satisfies one criterion of what an argument for a claim must be, since every sentence in a_1 has an immediate justification. a_1 would satisfy the account of an argument in ABA *if* that account were relaxed to allow the trees to be *infinite*. Yet we take the view that the sort of structure a_1 typifies, in which there is an infinite path of support, should not represent a possible pattern of dependency in an argument. This is because a chain of justifications must end somewhere: one cannot pass the buck forever. Thus, the definition of an argument in ABA, and in other standard approaches to structured argumentation, rightly excludes such infinite structures from being arguments.

a_2–a_4 all do conform to the definition of argument in ABA. However, as we now describe, a_2 has a kind of circularity which should be excluded; we then show that a_3 has a further kind of redundancy which also ought to be disallowed. Finally, we turn to the extensions in Fig. 3 and note that A has a generalized form of the problem with a_3.

In a_2 there is a circular dependency of a sentence, p, on itself (indirectly, through q): arguably, this ought not to be allowed as a representation of the way in which sentences are supported. It represents a situation in which the justification

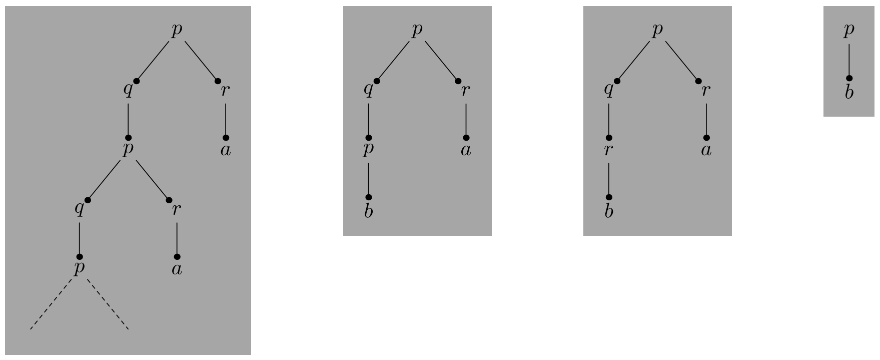


Fig. 2. Trees a_1–a_4 (left to right). a_1 does not qualify as an argument in ABA; a_2–a_4 do (see the definition in §2).

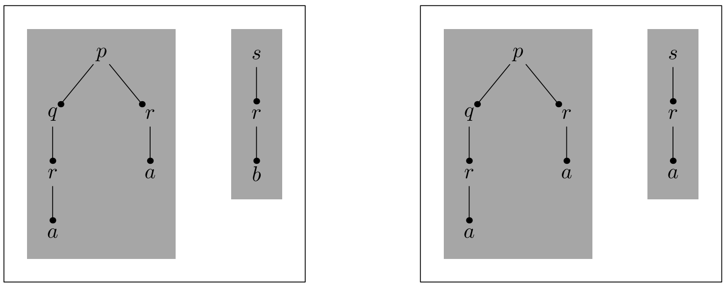


Fig. 3. Two extensions, A (left) and B (right), for Example 2.

for a given belief comes partly from itself, a notion which we regard as incoherent; the situation in which an agent attempts to justify a sentence on the grounds of that sentence itself is also rhetorically flawed. We therefore think that the definition of an argument should exclude the kind of dependency shown by a_2; in this way we will in §4 follow others such as [29], who have defined arguments so as to eliminate such dependency. (Note that in the case of ABA, where $\mathcal{R}$ is finite, then a tree in which there is an infinite path must also involve this type of circular dependency.)

Consider now a_3. Here, a sentence, r, is proved in two different ways. Even if neither proof involves a circularity (as in a_2), the argument can be deemed to be redundant and also to lead to inefficiency, since it supports the same sentence in two different ways at different points of the argument, using a and b as different reasons for the same conclusion. (It is important to note that this is different from the phenomenon of 'aggregation', in which a sentence is everywhere justified jointly by several reasons; aggregation is an important topic not addressed in ABA or other forms of structured argumentation, which we do not address in the current work either.) We therefore exclude the possibility of such redundant arguments.

The following definition formally captures the types of problem discussed for a_2 and a_3 from Example 2.

Definition 3.1. An argument a is *circular* if it contains a directed path from a node n labelled by some $s \in \mathcal{L}$ to another node n' labelled by s. An argument a is *flabby* if it is non-circular and there are different nodes n, n' labelled by $s \in \mathcal{L}$ such that the children of n are labelled by different members of $\mathcal{L} \cup \{\top\}$ from the children of n'. ⌟

According to this definition, in Example 2 a_2 is circular (and not flabby), a_3 is flabby (and not circular), and a_4 is neither circular nor flabby.

The sort of redundancy shown by a_3 can be found also in sets of arguments, as in A in Fig. 3. The same considerations in favour of ruling out flabby arguments also suggest ruling out extensions such as A, in which a sentence is supported in different ways in two different arguments. Indeed, extensions should represent dialectically coherent positions; if we rule out a reading whereby r is supported by an aggregation of the reasons a and b, then it should be supported either by a exclusively or b exclusively, as in B in Fig. 3 (for the case of support by a). Further, just as there can be a form of circularity internal to arguments, there might also be extensions in which, though no individual argument is circular, there are arguments a and b such that s depends on t in a, but t depends on s in b. The following definition captures these undesirable forms of extension.

Definition 3.2. An extension A is *bloated* if there are arguments $a, b \in A$ (possibly $a = b$) containing nodes n_a and n_b, where $label(n_a) = label(n_b)$ but $labels(children(n_a)) \neq labels(children(n_b))$. ⌟

According to this definition, A in Example 2 is bloated, but B is not.

The relations between the three concepts defined in this section are set out in the following.

Theorem 3.3. *Let* A *be a set of arguments. If* A *contains a circular or flabby argument, then* A *is bloated.*

Proof. If $a \in A$ is flabby, then it is trivial to show that A bloated.

Assume $a \in A$ is circular. Then there is a path in a from some node n_0 to some node n_l, both labelled by s. Let $(n_0, \ldots, n_m)$, for $0 < m \leqslant l$ be the smallest initial sequence of $(n_0, \ldots, n_l)$ ending in a node labelled by s, and let $(s_0, \ldots, s_m)$ be the corresponding sequence of labels of those nodes (so s_0 and s_m are both s). Then we consider in a the longest path

$$(n_0, \ldots, n_m, n_{m+1}, \ldots, n_{m+(m-1)}, n_{2m}, \ldots, n_{Km+I}),$$

such that the label of node n_{km+i} ($i < m$ and $0 \leqslant km + i \leqslant Km + I$) is identical to that of n_i. This path cannot be infinite, since arguments are finite. But then node n_{Km+I} must have differently-labelled children from node n_{m+I}, and thus A is bloated. □

The set A from Example 2 shows that the 'only if' direction of Theorem 3.3 fails to hold, namely a set of arguments may be bloated but contain no circular or flabby arguments.

Is the sort of circular dependency shown in a_2 exhibited in argumentation frameworks which represent real-world domains? In previous work, the current authors have worked on several practical applications of structured argumentation [8,18,19,30] to domains including medicine and law. We conducted an analysis of the argumentation frameworks for these applications,[5] which showed that none of them contained rules which allow the construction of cyclical arguments like a_2. (The largest of the frameworks we analyzed contained 11,147 rules.) This is minor evidence that the theoretical arguments for the *exclusion* of such structures from being arguments is also supported by the absence of such arguments from practical applications. However, it must also be noted that all of the argumentation frameworks in question were constructed by a single researcher or small group of researchers, trained in argumentation theory. Where applications involve disparately constructed knowledge bases, and where any cycles which may exist cannot be eliminated by a detailed revision and repair of the combined base, then the need for definitions of argument which themselves constrain the arguments constructed to be acyclic will become pressing.

Flabbiness and bloatedness, by contrast, do exist in real-world argumentation frameworks. All of the frameworks mentioned above contain many possibilities for the construction of flabby arguments, and this means that bloated extensions are also possible. Thus, there is a pragmatic justification for excluding the redundancy shown by many tree-based approaches to structured argumentation; this justification is seen even in argumentation frameworks in which the knowledge has been carefully represented.

We will show that circularity, flabbiness and bloatedness can be removed at a stroke by switching from a conception of arguments as trees to graphs, and by moving from a conception of an acceptable dialectical position as a set of arguments again to graphs.

4. Argument graphs

Argument graphs, which we introduce in the present section, serve two fundamental, related roles. First, they correspond to standard arguments: structures supporting a single, specific claim. Secondly, they correspond to the extensions used in defining the semantics of argumentation frameworks, and which are an overall representation of an acceptable dialectical position of an agent. Accordingly, in §4.1 below we initially treat argument graphs in their first role, and then in §4.3 in their second (where the emphasis will be on new forms of semantics defined in terms of argument graphs). An interlude, in §4.2, on the relations between argument graphs and forms of minimality fits between these two treatments. Recall, as stated in §2, that except where noted we are presuming a fixed ABA framework $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$.

In defining argument graphs below and in the rest of the paper we adopt the following notation. Where G is a graph, we sometimes use $v(G)$ to denote its vertices, and $e(G)$ to denote its edges (thus $e(G) \subseteq v(G) \times v(G)$). A *sink* vertex has no outgoing edges; we write the sink vertices of the directed graph G as $sinks(G)$. A *source* vertex is a vertex with no incoming edges.

Definition 4.1. An *argument graph* G is a directed, acyclic graph where $v(G) \subseteq \mathcal{L}$ and for all $s \in v(G)$:

i. if $s \in \mathcal{A}$, then $s \in sinks(G)$;
ii. if $s \notin \mathcal{A}$, then there is a rule $(s \leftarrow s_1, \ldots, s_m) \in \mathcal{R}$ such that there is an edge (s, s') in $e(G)$ iff $s' \in \{s_1, \ldots, s_m\}$.

Where G is an argument graph, the *support* of G, written $support(G)$, is $v(G) \cap \mathcal{A}$.

An argument graph G is said to be *focused* iff it has a unique source, called the *claim* of G and represented as $claim(G)$. ⌟

[5] We thank an anonymous reviewer for suggesting this.

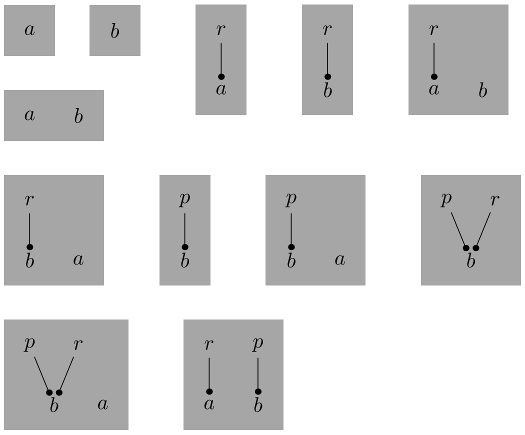


Fig. 4. Selection of argument graphs for Example 2.

Note that vertices of argument graphs *are* sentences, whereas nodes of standard arguments in ABA, as given in §2, are *labelled by* sentences (or $\top$). Note also that we overload the terms 'support' and 'claim', defined with similar intentions for both tree-based arguments and argument graphs. Fig. 4 shows all argument graphs that can be obtained from the ABA framework in Example 2, if the rules in that framework are restricted to:

$$\begin{aligned}\mathcal{R} = \{ & p \leftarrow b, \\ & r \leftarrow a, \\ & r \leftarrow b \}.\end{aligned}$$

Here, for example, the top, left-most argument with support a is focused, with claim a, whereas the top, right-most argument with support $\{a, b\}$ is not focused, as it has two sources, r and b. The bottom-right argument graph, $G_{p,r}$, can be written as $(\{p, r, a, b\}, \{(p, b), (r, a)\})$. This makes the nature of $v(G_{p,r})$ and $e(G_{p,r})$ as a sets of sentences and pairs, respectively, explicit.

Note that, in Fig. 4 and throughout the paper, in visualizing argument graphs, we follow the same conventions, given in Example 1, as for tree-based arguments (e.g., the direction of an edge is represented by the relative vertical position of its nodes as well as disambiguated by a dot).

4.1. Argument graphs and tree-based arguments

In this section, we show how the notion of an argument graph addresses some of the issues raised in §3 and how it relates to the original notion of ABA arguments as trees.

Argument graphs can be seen as representing only those tree-based arguments without the undesirable properties of the kinds identified in §3. The required notion of *representation* is given in the following.

Definition 4.2. Let G be an argument graph, and a a tree-based argument. We say that a is *represented in* G if there is a function $f : (nodes(\mathsf{a}) \setminus \{n \mid n \in nodes(\mathsf{a}) \wedge label(n) = \top\}) \to v(G)$ mapping nodes of a not labelled by $\top$ to nodes of G such that, where $n \in (nodes(\mathsf{a}) \setminus \{n \mid n \in nodes(\mathsf{a}) \wedge label(n) = \top\})$:

- $f(n) = label(n)$;
- if $f(n) = s$, then $labels(\{n' \mid n' \in children(n)\}) \setminus \{\top\} = \{s' \mid (s, s') \in e(G)\}$. ⌟

This notion is illustrated below.

Example 3. Consider the ABA framework:

$$\mathcal{L} = \{ p, q, r, s, t, x, a \}$$

$$\begin{aligned}\mathcal{R} = \{ & p \leftarrow q, r, \\ & q \leftarrow s, \\ & r \leftarrow s, a, \\ & s \leftarrow,\end{aligned}$$

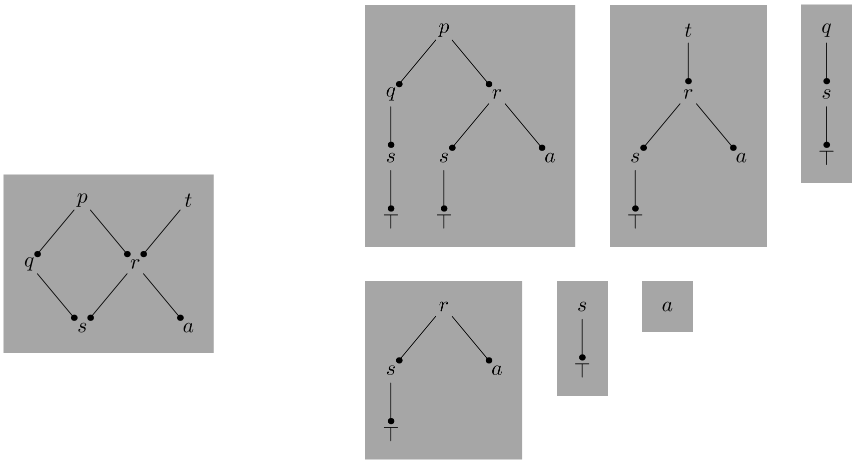


Fig. 5. Argument graph (left) and the arguments represented in it (right), for Example 3.

$$t \leftarrow r \,\}$$

$$\mathcal{A} = \{\, a \,\}$$

$$\bar{a} = x$$

Fig. 5 shows an argument graph and the arguments it represents. ⌟

Fig. 5 demonstrates the concision in representation afforded by argument graphs, compared with tree-based arguments. Indeed, the following result shows that argument graphs are a combined representation of several tree-based arguments—one argument for each node in the argument graphs, with the claim the node in question.

Theorem 4.3. *Let G be an argument graph. For each $s \in v(G)$, there is an argument* a *such that $claim(\mathrm{a}) = s$, $support(\mathrm{a}) \subseteq support(G)$ and* a *is represented in G.*

Proof. We generate an argument from s and G in the following way. First, create a new node n_r labelled by s (n_r has depth 0 in the argument). Then, starting at $i = 0$, for each node n' at depth i, where n' is labelled by s', for each s'' such that $(s', s'') \in e(G)$ create a new child n'' of n' in a labelled by s''. Then increment i until there are no nodes of depth i. Since there are no cycles in G, this process terminates.

Let a be the resulting argument; to show that a is represented in G, let $f : nodes(\mathrm{a}) \to v(G)$ be such that $f(n) = label(n)$. According to Definition 4.2 we just need to show that if $f(n) = s$ for any $n \in nodes(\mathrm{a})$, then $labels(\{n' \mid n' \in children(n)\}) \setminus \{\top\} = \{s' \mid (s, s') \in e(G)\}$. This is true by construction.

Since a is represented in G, it is immediate that $support(\mathrm{a}) \subseteq support(G)$. □

We have seen in §3 that tree-based arguments may be circular or flabby in general. The following theorem shows that the desirable properties of non-circularity and non-flabbiness hold for tree-based arguments represented in argument graphs.

Theorem 4.4. *Let G be an argument graph, and* a *an argument represented in G. Then* a *is neither circular nor flabby.*

Proof. Let a be represented in G. Non-circularity is easy to show, given the non-circularity of G. Assume a is flabby. Then there are $n_1, n_2 \in nodes(\mathrm{a})$ such that $n_1 \neq n_2$, $label(n_1) = label(n_2)$, and $labels(children(n_1)) \neq labels(children(n_2))$. But from $label(n_1) = label(n_2)$ (from the second bullet in Definition 4.2) we know that $labels(children(n_1)) = labels(children(n_2))$. Contradiction. □

To investigate the converse direction to Theorem 4.4, i.e., whether each non-circular, non-flabby argument is represented in some argument graph, we first define a mapping from tree-based arguments to argument graphs, as follows:

Definition 4.5. Let a be a tree-based argument. A focused argument graph G is a *graphical conversion of* a if:

- $claim(\mathrm{a}) = claim(G)$;
- if $(s, s') \in e(G)$, then there are $n, n' \in nodes(\mathrm{a})$ such that $n' \in children(n)$, $label(n) = s$ and $label(n') = s'$.

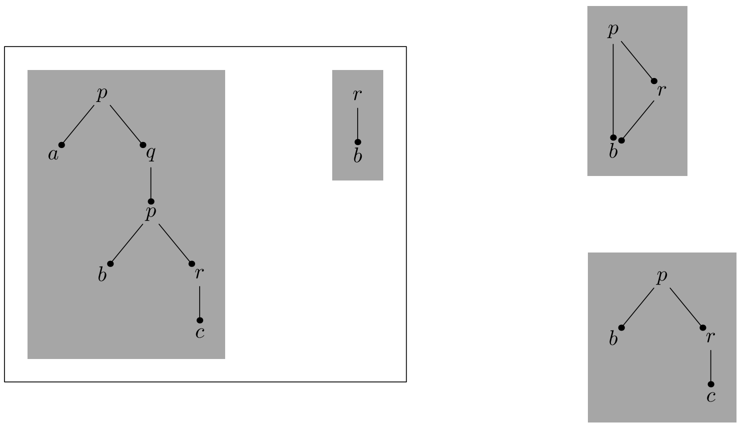


Fig. 6. A set of arguments A (left) for Example 4, with two argument graphs, G_1 (top) and G_2 (bottom), each of which is a graphical conversion of A (right).

Let A be a set of tree-based arguments. An argument graph G is a *graphical conversion of* A if:

- $claims(\mathsf{A}) \subseteq v(G)$;
- if $(s, s') \in e(G)$, there is $\mathsf{a} \in \mathsf{A}$ and $n, n' \in nodes(\mathsf{a})$ such that $n' \in children(n)$, $label(n) = s$ and $label(n') = s'$. ⌟

This notion of graphical conversion is illustrated in the following example.

Example 4. Consider the ABA framework:

$$\begin{aligned}
&\mathcal{L} = \{\, p, q, r, z, a, b, c \,\}\\
&\mathcal{R} = \{\, p \leftarrow a, q,\\
&\qquad p \leftarrow b, r,\\
&\qquad q \leftarrow p,\\
&\qquad r \leftarrow b,\\
&\qquad r \leftarrow c \,\}\\
&\mathcal{A} = \{\, a, b, c \,\}\\
&\bar{a} = z,\ \bar{b} = z,\ \bar{c} = z
\end{aligned}$$

Fig. 6 shows a set of arguments A with two graphical conversions G_1 and G_2 of that set. Note that $claims(\mathsf{A})$ is $\{p, r\}$, and $\{p, r\} \subseteq v(G_1)$, $\{p, r\} \subseteq v(G_2)$, as Definition 4.5 requires. Further, for any edge in either G_1 or G_2, there is a similarly-labelled edge in one of the arguments in A. ⌟

A graphical conversion of an argument a is a focused argument graph G with the same claim, which can be thought of as pruning the argument in such a way as to remove circularity and flabbiness. Similarly, a graphical conversion of a set of arguments A is an argument graph that retains the claims of arguments in A as nodes, but prunes those arguments in such a way as to remove bloatedness (and thus also, given Theorem 3.3, circularity and flabbiness). As a consequence, the graphical conversion of a non-circular, non-flabby argument and the graphical conversion of a non-bloated set of arguments are guaranteed to be unique and to have the same claims and the same supports as the original (sets of) arguments, as we now prove.

Theorem 4.6. *(i) Let* a *be a non-circular, non-flabby argument. Then there is a unique graphical conversion G of* a *which is a focused argument graph with $claim(\mathsf{a}) = claim(G)$ and $support(\mathsf{a}) = support(G)$, such that* a *is represented in G. (ii) Let* A *be a non-bloated set of arguments. Then there is a unique graphical conversion G of* A *with $claims(\mathsf{A}) \subseteq v(G)$ and $\bigcup_{\mathsf{a} \in \mathsf{A}} support(\mathsf{a}) = support(G)$ and each argument in* A *is represented in G.* ⌟

Note that for any non-circular, non-flabby argument a there may, in fact, be many different argument graphs (not necessarily focused) in which a is represented; this occurs when there are argument graphs G' where a graphical conversion G of a is such that G is a sub-graph of G', as illustrated next.

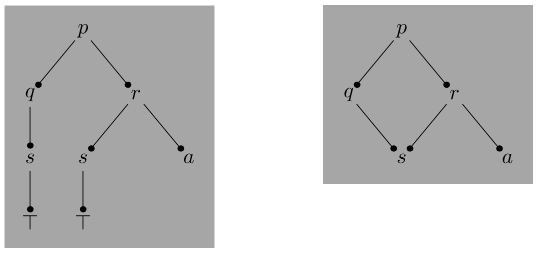


Fig. 7. A non-circular, non-flabby argument (left) and its graphical conversion (right), for Example 3.

Example 4 *(Continued).* Fig. 7 shows (left) an argument (from Fig. 5) with its (focused) graphical conversion (right). The left argument is also represented in the (non-focused) argument graph in Fig. 5 (left). ⌟

Theorems 4.4 and 4.6 together mean that all and only those tree-based arguments that are both non-circular and non-flabby are represented in argument graphs. Directly from these results, it is easy to see that there is an analogous relationship of equivalence between non-circular, non-flabby arguments and tree-based arguments in general.

Theorem 4.7. *Let $s \in \mathcal{L}$ and $A \subseteq \mathcal{A}$. (i) For every non-circular, non-flabby tree-based argument for s supported by A there exists a tree-based argument for s supported by A. (ii) For every tree-based argument for s supported by A there exists a non-circular, non-flabby tree-based argument for s supported by $A' \subseteq A$.* ⌟

Theorems 4.4, 4.6 and 4.7 easily let us show the following, which establishes the desired result: argument graphs are, in at least one important sense, equivalent to the tree-based arguments in standard ABA.

Corollary 4.8. *Let $s \in \mathcal{L}$ and $A \subseteq \mathcal{A}$. (i) For every focused argument graph G with claim s supported by A, there exists an argument* a *with claim s supported by A. (ii) For every argument* a *with claim s supported by A, there exists a focused argument graph G with claim s supported by $A' \subseteq A$.*

Proof. (i) Let G be a focused argument graph, with $claim(G) = s$ and support A. Then by Theorem 4.3 there is an argument a represented in G, such that $claim(\mathrm{a}) = s$ and $support(\mathrm{a}) \subseteq support(G)$. It is easy to see that at least one such a must be such that $support(\mathrm{a}) = support(G)$.

(ii) Let a be an argument with $claim(\mathrm{a}) = s$ and $support(\mathrm{a}) = A$. Then by Theorem 4.7(ii) there is a non-circular, non-flabby argument a' with the same claim, and support A', where $A' \subseteq A$. Then by Theorem 4.6 there is an argument graph G with $claim(G) = s$ and $support(G) \subseteq A$. □

Corollary 4.8 indicates that it is possible, when determining whether a *sentence* is acceptable according to the various standard extension-based semantics of ABA, to restrict attention to argument graphs. Indeed, in §4.3 we redefine ABA semantics of admissible and grounded extensions in terms of argument graphs. Yet although the corollary shows that there is a correspondence in terms of the existence of arguments for a given claim, one of the main strengths of using structured argumentation is that the detailed justification for such claims is presented. As we argued in §3, using trees as representations for arguments typically admits circularity, flabbiness and bloatedness, and all of these should be excluded, something which is achieved by the use of argument graphs. We therefore think that there are strong conceptual reasons for preferring them to tree-based arguments. Other gains, in terms of the dispute derivations and efficiency, are discussed in later sections.

First though, we consider the relationships between argument graphs and two restricted types of tree-based arguments.

4.2. Argument graphs and two forms of minimality

To avoid redundancies in arguments, of the form illustrated in Example 2, several existing approaches to structured argumentation, e.g., logic-based argumentation [5] and DeLP [20], impose forms of minimality on the support of arguments. In this section we consider two forms of minimality, in relation to argument graphs.

Definition 4.9. An argument a is *rule-minimal* iff for any two nodes n, n' in a labelled by the same $s \in \mathcal{L}$ the children of n and n' are labelled by the same elements of $\mathcal{L} \cup \{\top\}$. ⌟

All arguments shown to the right in Fig. 5 are rule-minimal; arguments a_2 and a_3 from Fig. 2 are not.

An alternative way to impose absence of circularity and flabbiness in an argument is to require that it be rule-minimal.

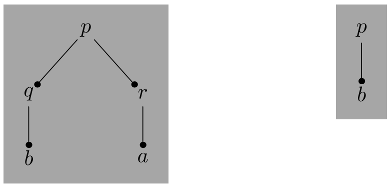


Fig. 8. Argument graphs from Example 5: G_1 (left) and G_2 (right).

Theorem 4.10. *An argument is rule-minimal iff it is neither circular nor flabby.*

Proof. Let a be an argument. First assume a is rule-minimal. By Definitions 3.2 and 4.9, {a} is not bloated. Thus by the contrapositive of Theorem 3.3, a is neither circular nor flabby.

For the other direction, suppose that a is neither flabby nor circular. Assume for contradiction that it is not rule-minimal. Then there are nodes n and n' labelled by some s such that the children of n and n' are differently labelled. n and n' cannot be on the same path in a, for then a would be circular. So they are not; but then a must be flabby. Contradiction. □

Thus, results from §4.1 which make reference to non-circular, non-flabby arguments, are equivalent to analogous theorems for rule-minimal arguments: Theorem 4.4, for example, can be rephrased as stating that if a is represented in an argument graph, then a must be rule-minimal.Similar rephrasings of Theorems 4.6, 4.7 and Corollary 4.8 also hold.

Given the relationship between rule-minimal arguments and argument graphs, it should be asked what the advantages or disadvantages are of using one over the other. First, although a focused argument graph can be seen as equivalent to a single rule-minimal, tree-based argument, argument graphs also function for us as representations of a number of different arguments operating dialectically as one: they play the role of extensions, as we will see more clearly in §4.3 when discussing argument-graph semantics. Bloated extensions are not excluded by confining ourselves to rule-minimal arguments—as we saw in Example 2, A is a set of rule-minimal arguments but is bloated. Secondly, argument graphs are inherently more compact representations: the possible presence of different nodes in a rule-minimal, tree-based argument, labelled by the same sentence, is wasteful.

While focused argument graphs correspond to rule-minimal arguments and remove several forms of redundancy, they may still contain redundancies in their support, as illustrated by the following example.

Example 5. Consider the ABA framework in Example 2 but with $q \leftarrow r$ in $\mathcal{R}$ replaced by $q \leftarrow b$. Shown in Fig. 8 are argument graphs G_1 and G_2 for p, supported by $\{a, b\}$ and $\{b\}$ respectively. Here, the support of G_1 is non-minimal, in that $support(G_2) \subset support(G_1)$. ⌟

Thus, an alternative notion of minimality for arguments is obtained by requiring that their supports are minimal. The same idea can be applied to argument graphs, as follows.

Definition 4.11. A focused argument graph G is *support-minimal* iff there is no focused argument graph G' with $claim(G) = claim(G')$ such that $support(G') \subset support(G)$. ⌟

Example 5 shows that argument graphs, though representing only rule-minimal arguments, may not be support-minimal. (In relation to this, see the remarks in Section 7 on the work of Besnard and Hunter [4] and García and Simari [20].)

An analogous notion of support-minimal tree-based arguments can be easily defined. It is also true that, where some argument a is support-minimal, it does not need to be rule-minimal—and so does not have to correspond directly to an argument graph. To see this, consider the following simple example.

Example 6. Consider the ABA framework with

$$\begin{aligned}
\mathcal{L} &= \{\, p, x, a \,\} \\
\mathcal{R} &= \{\, p \leftarrow p, \\
&\qquad p \leftarrow a \,\} \\
\mathcal{A} &= \{\, a \,\} \\
&\quad \bar{a} = x
\end{aligned}$$

In Fig. 9, arguments a_1 and a_2 are for p and are support-minimal, being supported by $\{a\}$, but only a_2 is rule-minimal. ⌟

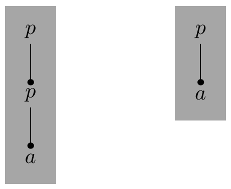


Fig. 9. Arguments for Example 6. Both a_1 (left) and a_2 (right) are support-minimal, but only a_2 is rule-minimal.

Thus, support-minimality does not guarantee non-circular and non-flabby arguments. Moreover, identifying support-minimal arguments may be computationally demanding. Indeed, whereas the notion of support-minimality is 'global', in that to check whether a focused argument graph is support-minimal it may need to be compared with all other focused argument graphs for the same claim, the notion of rule-minimality is 'local', in that to check whether an argument is rule-minimal all that is required is a syntactic check of the argument.

4.3. Argument graph semantics

As noted earlier, argument graphs can be used to replace sets of tree-based arguments (extensions) as the basic unit of semantics. In discussing extensions in §3, we noted that *bloatedness* was an undesirable property. The following theorem shows that argument graphs as the analogues of extensions avoid it.

Theorem 4.12. *Let G be an argument graph, and A the set of arguments represented in G. Then A is not bloated.*

Proof. We must show that there are no two arguments a and b in A such that $claim(\mathsf{a}) = claim(\mathsf{b})$. Suppose, for contradiction, that a and b are two such arguments. Then a and b are identical up to some depth $i > 0$, but there is some node n_{a} of a, and node n_{b} of b, both of depth i, such that $label(n_{\mathsf{a}}) = label(n_{\mathsf{b}})$, but

$$labels(children(n_{\mathsf{a}})) \neq labels(children(n_{\mathsf{b}}))$$

But by Definition 4.2, this means that, where a' is the argument given by the subtree of a rooted at n_{a}, and b' is the argument given by the subtree of b rooted at n_{b}, then

$$\{s' \mid (claim(\mathsf{a}'), s') \in e(G)\} \neq \{s' \mid (claim(\mathsf{b}'), s') \in e(G)\}$$

But this is impossible, by Definition 4.1, since $claim(\mathsf{a}') = claim(\mathsf{b}')$. Contradiction. Thus there are no such a and b and A is not bloated. □

In moving to argument graphs as a representation of the relations of rational support between sentences according to an agent, we need to define corresponding versions of existing, extension-based, argumentation semantics. To do so, though, we first need to redefine the notion of attack, as follows:

Definition 4.13. Let G, G' be two argument graphs. Then G *attacks* G', written $G \rightsquigarrow G'$, if there is $a \in v(G')$ such that $\bar{a} = s$, for some $s \in v(G)$. ⌟

Note that, since an argument graph G represents the relations of rational support holding amongst sentences, then the agent has an argument for any $s \in \mathcal{L}$ if $s \in v(G)$. Thus, it is appropriate to define a relation of attack between argument graphs, $G \rightsquigarrow G'$, allowing the sentence in G that is contrary of the attacked assumption in G' to feature *anywhere* in G. This contrasts with the definition of attack in tree-based ABA, in which only the claim—i.e., the root—of the argument is relevant. The following example illustrates the notion of attack between argument graphs and how it differs from the standard ABA notion of attack between arguments.

Example 7. Consider the ABA framework:

$$\begin{aligned}
\mathcal{L} &= \{\, p, x, y, z, a, b, c \,\} \\
\mathcal{R} &= \{\, p \leftarrow a, \\
&\qquad p \leftarrow b, \\
&\qquad x \leftarrow y, \\
&\qquad y \leftarrow c \,\} \\
\mathcal{A} &= \{\, a, b, c \,\} \\
\bar{a} &= y,\ \bar{b} = z,\ \bar{c} = y
\end{aligned}$$

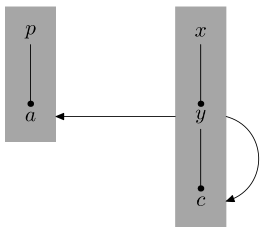


Fig. 10. Attacks in Example 7. The argument graph G_1 is on the left, and G_2 is on the right. $G_2 \rightsquigarrow G_1$ and $G_2 \rightsquigarrow G_2$.

Fig. 10 shows the attacks between two argument graphs for this framework. To the left, there is a focused argument graph G_1 with claim p, and to the right, a focused argument graph G_2 with claim x. Since $a \in v(G_1)$ and $\bar{a} = y$, with $y \in v(G_2)$, then $G_2 \rightsquigarrow G_1$. Similarly, since $c \in v(G_2)$ and $\bar{c} = y$, then G_2 attacks itself: $G_2 \rightsquigarrow G_2$. Note that if the graphs were interpreted as tree-based arguments, then neither of these attacks would be present, since they both stem from an internal node (y is not the claim of G_2). ⌟

Using the notion of attack between argument graphs, we can proceed to analogues of the standard extension-based semantics. Some extension-based semantics make use of the relation of subset inclusion, $\subseteq$, in order to impose a maximality or minimality requirement on the extensions. In moving to argument graphs, we must use an analogous relation—that of *subgraph*: where G and G' are graphs (including argument graphs), then G is a *subgraph* of G' (written $G \subseteq G'$) iff:

- $v(G) \subseteq v(G')$; and
- $e(G)$ is the restriction of $e(G')$ to $v(G)$, i.e.,

$$e(G) = e(G') \cap (v(G) \times v(G)).$$

Moreover, G is a *proper subgraph* of G' (written $G \subset G'$) iff $G \subseteq G'$ and $G \neq G'$. Finally, we use $\Diamond$ to represent the empty graph $(\emptyset, \emptyset)$.

Recall the definition of a characteristic function for an abstract argumentation framework (presented in §2). In defining our semantics for argument graphs, we will make use of the idea of an *argument graph characteristic function*, applied here to sets of argument graphs instead of sets of arguments. We will make use of the following notions.

Definition 4.14. Let G be an argument graph. We define *rules*(G) to be

$$\{s \leftarrow s_1, \ldots, s_m \mid (s \leftarrow s_1, \ldots, s_m) \in \mathcal{R} \wedge s \in v(G)) \wedge \forall s'((s, s') \in e(G) \leftrightarrow s' \in \{s_1, \ldots, s_m\})\}.$$

Let $\mathsf{R} \subseteq \mathcal{R}$. We will say that R is *rule-consistent* iff there are no $s \leftarrow s_1, \ldots, s_m$, $s \leftarrow s'_1, \ldots, s'_k$ in R such that $\{s_1, \ldots, s_m\} \neq \{s'_1, \ldots, s'_k\}$. Further, R is *maximally rule-consistent* iff R is $\subseteq$-maximally rule-consistent. ⌟

Thus, *rules*(G) is, informally, the set of rules used in the construction of G. In Fig. 10, for Example 7, *rules*$(G_1) = \{p \leftarrow a\}$ and *rules*$(G_2) = \{x \leftarrow y,\ y \leftarrow c\}$. In this example, there are two maximally rule-consistent sets, $\mathsf{R}_1 = \{p \leftarrow a,\ x \leftarrow y,\ y \leftarrow c\}$ and $\mathsf{R}_2 = \{p \leftarrow b,\ x \leftarrow y,\ y \leftarrow c\}$.

We now give the definition of argument graph characteristic function. This is analogous to the definition of the characteristic function of an abstract argumentation framework (see §2), in that it gives a maximal argument graph which is defended; but we parameterize the function on maximally rule-consistent sets R to capture the choice of rules underlying the construction of an argument graph. Formally:

Definition 4.15. Let $\mathsf{R} \subseteq \mathcal{R}$ be maximally rule-consistent, and let G be the set of all argument graphs G such that *rules*$(G) \subseteq \mathsf{R}$. The *argument graph characteristic function w.r.t.* R is the function $f_{\mathsf{R}} : \mathsf{G} \to \mathsf{G}$ such that for all argument graphs G where *rules*$(G) \subseteq \mathsf{R}$, $f_{\mathsf{R}}(G)$ is the $\subseteq$-maximal argument graph G' such that:

i. *rules*$(G') \subseteq \mathsf{R}$;
ii. for any argument graph G^*, if $G^* \rightsquigarrow G'$ then $G \rightsquigarrow G^*$. ⌟

As shown above, for an argument graph G there may in general be several maximally rule-consistent R such that *rules*$(G) \subseteq \mathsf{R}$. Thus, different argument graph characteristic functions can be applied to G to yield (possibly) $\subseteq$-larger argument graphs. To illustrate this, consider again Example 7, and let G_b be the argument graph just containing a single node b (so that $v(G_b) = \{b\}$ and $e(G_b) = \emptyset$). Then, for $\mathsf{R}_1 = \{p \leftarrow a,\ x \leftarrow y,\ y \leftarrow c\}$ and $\mathsf{R}_2 = \{p \leftarrow b,\ x \leftarrow y,\ y \leftarrow c\}$ given above, it is plain that $f_{\mathsf{R}_1}(G_b) = G_b$, and $f_{\mathsf{R}_2}(G_b)$ is $G_{p,b}$, where $v(G_{p,b}) = \{p, b\}$ and $e(G_{p,b}) = \{(p, b)\}$.

The following provides a property of argument graph characteristic functions which we use frequently.

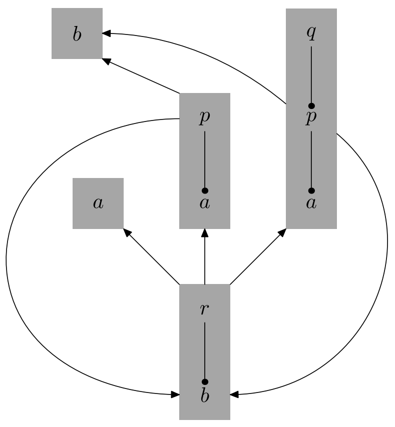


Fig. 11. Argument graphs and attacks for Example 8.

Theorem 4.16. *Any argument graph characteristic function f_{R} has a least fixed point equal to $f_{\mathsf{R}}^{\omega}(\lozenge)$.* ⌟

Now, properties of the argument graph characteristic function can then be used to define semantics for argument graphs, in a way analogous to the relationship between characteristic functions and semantics in abstract argumentation.

Definition 4.17. Let G be an argument graph.

- G is *conflict-free* iff it is not the case that $G \rightsquigarrow G$.
- G is *admissible* iff it is conflict-free and for all argument graph characteristic functions f_{R} such that $rules(G) \subseteq \mathsf{R}$, we have $G \subseteq f_{\mathsf{R}}(G)$.
- G is *complete* iff it is conflict-free and for all argument graph characteristic functions f_{R} such that $rules(G) \subseteq \mathsf{R}$, we have $G = f_{\mathsf{R}}(G)$.
- G is *grounded* iff it is conflict-free and for all argument graph characteristic functions f_{R} such that $rules(G) \subseteq \mathsf{R}$, G is the least fixed point of f_{R}. ⌟

These are increasingly strong: if G is grounded, it is complete; if G is complete, it is admissible; and if G is admissible, it is conflict-free.

Example 8. Consider the ABA framework:

$$\begin{aligned}
&\mathcal{L} = \{\, p, q, r, a, b \,\}\\
&\mathcal{R} = \{\, p \leftarrow a,\\
&\qquad q \leftarrow p,\\
&\qquad r \leftarrow b \,\}\\
&\mathcal{A} = \{\, a, b \,\}\\
&\bar{a} = r,\ \bar{b} = p
\end{aligned}$$

Note that here only one set of rules, $\mathcal{R}$ itself, is maximally rule-consistent. The argument graphs for this framework—without $\lozenge$—and attacks between them are shown in Fig. 11. Since there are no two argument graphs with the same claim in this example, we let G_x denote the argument graph whose claim is x (for any $x \in \mathcal{L}$).

All argument graphs here are conflict-free. G_p is admissible, since we have $G_p \subseteq f_{\mathcal{R}}(G_p)$ (in fact, $f_{\mathcal{R}}(G_p) = G_q$); also admissible are G_q itself, G_r, and the empty argument graph $\lozenge$. Since $f_{\mathcal{R}}(G_p) \neq G_p$, G_p is not complete. However, since $f_{\mathcal{R}}(G_q) = G_q$, $f_{\mathcal{R}}(G_r) = G_r$ and $f_{\mathcal{R}}(\lozenge) = \lozenge$, all of G_q, G_r and $\lozenge$ are complete. Finally, $\lozenge$ is the only grounded argument graph. ⌟

It might be asked whether the universal quantification over argument graph characteristic functions in Definition 4.17 is essential, or whether the definition could be weakened to existential quantification and be equivalent. For the admissible semantics, this is the case, as the following theorem shows.

Theorem 4.18. *Let G be conflict free. G is admissible iff for some argument graph characteristic function f_{R} such that $rules(G) \subseteq \mathsf{R}$, we have $G \subseteq f_{\mathsf{R}}(G)$.*

Proof. Evidently if G is an argument graph then there is some f_{R} such that $rules(G) \subseteq \mathsf{R}$; we therefore just need to show that if $G \subseteq f_{\mathsf{R}}(G)$ for some such function, then where $rules(G) \subseteq \mathsf{R}'$ for some $f_{\mathsf{R}'}$, we have $G \subseteq f_{\mathsf{R}'}(G)$.

Suppose for contradiction that there is R' such that $G \not\subseteq f_{\mathsf{R}'}(G)$. Then there must be $a \in v(G) \cap \mathcal{A}$ such that $a \notin f_{\mathsf{R}'}(G)$, which means that there must be some G^* such that $G^* \leadsto G$ but not $G \leadsto G^*$, with G^* attacking G at a. Yet then $a \notin f_{\mathsf{R}}(G)$, contradicting the fact that $G \subseteq f_{\mathsf{R}}(G)$. So there is no such maximally rule-consistent R'. □

By contrast, Example 7 affords an example of why the equivalence which Theorem 4.18 states does not hold for the complete or grounded semantics. To see this, consider again the (conflict-free) argument graph G_b such that $v(G_b) = \{b\}$ and $e(G_b) = \emptyset$, and the two maximally rule-consistent sets $\mathsf{R}_1 = \{p \leftarrow a,\ x \leftarrow y,\ y \leftarrow c\}$ and $\mathsf{R}_2 = \{p \leftarrow b,\ x \leftarrow y,\ y \leftarrow c\}$. Evidently it is true that $f_{\mathsf{R}_1}(G_b) = G_b$, so that there is *some* set R_1 for which G_b is a fixed point; but R_2 is also such that $rules(G_b) \subseteq \mathsf{R}_2$, and yet $f_{\mathsf{R}_2}(G_b) \neq G_b$.

The various semantics introduced earlier can be equivalently reformulated without making use of the argument graph characteristic function, in a style similar to that of the definition of the corresponding semantics in abstract argumentation, by virtue of the following result.

Theorem 4.19. *Let G be a conflict-free argument graph.*

i. *G is admissible iff for any argument graph G' such that $G' \leadsto G$, then $G \leadsto G'$.*
ii. *G is complete iff it is admissible and there is no argument graph G' such that $G \subset G'$ and for all argument graphs G^*, if $G^* \leadsto G'$, then $G \leadsto G^*$.*
iii. *If G is grounded, then it is $\subseteq$-minimally complete.*

Proof. Let G be conflict-free.

i. First suppose G is admissible, and that G' is such that $G' \leadsto G$. Pick any f_{R} such that $rules(G) \subseteq \mathsf{R}$. Then since G is admissible, $G \subseteq f_{\mathsf{R}}(G)$. But then $G' \leadsto f_{\mathsf{R}}(G)$, so that by the definition of f_{R}, $G \leadsto G'$.
   Suppose now that for any argument graph G' such that $G' \leadsto G$, then $G \leadsto G'$ (namely G defends itself). We must show that G is admissible. G is conflict-free, so that we need to show that for all argument graph characteristic functions f_{R} such that $rules(G) \subseteq \mathsf{R}$, $G \subseteq f_{\mathsf{R}}(G)$. Let f_{R} be any such function. That $G \subseteq f_{\mathsf{R}}(G)$ follows directly from Definition 4.15 and G defending itself.
ii. Suppose G is complete; then it is admissible. We must show that there is no G' such that $G \subset G'$ and for all G^*, if $G^* \leadsto G'$ then $G \leadsto G^*$. Suppose there is such a G', for contradiction, and let $f_{\mathsf{R}'}$ be such that $rules(G') \subseteq \mathsf{R}'$. By the definition of complete argument graph, and the fact that $rules(G) \subseteq rules(G')$, it must be that $G = f_{\mathsf{R}'}(G)$. But that violates the $\subseteq$ -maximality condition in Definition 4.15: contradiction. So there is no such G'.
   Suppose now that G is admissible and there is no G' such that $G \subset G'$ and for all G^*, if $G^* \leadsto G'$ then $G \leadsto G^*$. Let f_{R} be such that $rules(G) \subseteq \mathsf{R}$. By admissibility, $G \subseteq f_{\mathsf{R}}(G)$, so we only need to show that $f_{\mathsf{R}}(G) \subseteq G$. The result easily follows by Definition 4.15.
iii. Assume G is grounded. Plainly it is complete; we must show that it is $\subseteq$-minimally so. Suppose for contradiction there is $G^- \subset G$ such that G^- is complete. Then, by Definition 4.17, for all f_{R^-} such that $rules(G^-) \subseteq \mathsf{R}^-$, we have $G^- = f_{\mathsf{R}^-}(G^-)$. But then, since $G^- \subset G$, we have that $rules(G^-) \subset rules(G)$, so that for all f_{R} such that $rules(G) \subseteq \mathsf{R}$, $G^- = f_{\mathsf{R}}(G^-)$, i.e., G^- is a fixed point for all such f_{R}. This contradicts the groundedness of G. So there is no such G^-, and G is $\subseteq$-minimally complete. □

It is important to note here that the conditional (iii) of Theorem 4.19 cannot be strengthened to a biconditional, as we show next.

Example 9. Consider the ABA framework:

$$
\begin{aligned}
&\mathcal{L} = \{\, p, q, x, y, z, a, b, d, c, e \,\} \\
&\mathcal{R} = \{\, p \leftarrow a, \\
&\qquad\quad q \leftarrow p, b, \\
&\qquad\quad q \leftarrow p, d, \\
&\qquad\quad y \leftarrow c, \\
&\qquad\quad z \leftarrow e \,\} \\
&\mathcal{A} = \{\, a, b, c, d, e \,\} \\
&\bar{a} = x,\ \bar{b} = y,\ \bar{c} = q,\ \bar{d} = z,\ \bar{e} = p
\end{aligned}
$$

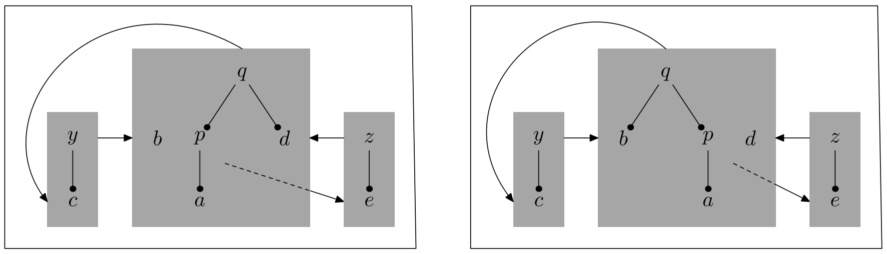


Fig. 12. Argument graphs and attacks for Example 9.

Now consider Fig. 12. First consider the argument graphs in the left-hand box. The large, central argument graph G_1 (containing q) defends itself from the argument graph containing y (G_2, left) and the argument graph containing z (G_3, right). Since these are the possible attacks on G_1, G_1 is admissible. Then it is also complete. Is it grounded? Yes, since $rules(G_1)$=$\{p \leftarrow a,\ q \leftarrow p, d\}$; it is not hard to see that for any f_{R} such that $rules(G_1) \subseteq \mathsf{R}$, G_1 is the least fixed point of f_{R}; and, as ensured by Theorem 4.19, G_1 is $\subseteq$-minimally complete.

Now consider the argument graphs in the right-hand box. Let G_4 be the central argument graph (containing q), with G_5 the argument graph to its left containing y, and G_6 the argument graph to its right. Plainly, $v(G_1) = v(G_4)$; the difference between them is in the rule used to support q. As before, G_4 is admissible, and it is also $\subseteq$-minimally complete. However, it is not grounded: for consider $\mathsf{R} = \{p \leftarrow a,\ q \leftarrow b, p,\ y \leftarrow c,\ z \leftarrow e\}$. This is the only maximally rule-consistent set of rules such that $rules(G_4) \subseteq \mathsf{R}$. Now:

$$f_{\mathsf{R}}(\lozenge) = (\{p, a\}, \{(p, a)\})$$
$$f^2_{\mathsf{R}}(\lozenge) = (\{p, a, d\}, \{(p, a)\})$$
$$f^3_{\mathsf{R}}(\lozenge) = f^2_{\mathsf{R}}(\lozenge)$$

Let $G' = (\{p, a, d\}, \{(p, a)\})$. Then $G' \neq G_4$ is the least fixed point of all f_{R} such that $rules(G_4) \subseteq \mathsf{R}$ (as we have seen there is only one such R), so that G_4 cannot be grounded. Note that G' itself is not grounded, since it is not the least fixed point of all $f_{\mathsf{R}'}$ such that $rules(G') \subseteq \mathsf{R}'$, e.g. for $\mathsf{R}' = rules(G_1)$ for G_1 considered earlier.

The fact that G_1 is grounded and G_4 is not can be intuitively understood with reference to the way the argument graph characteristic function operates with respect to the two sets of rules. q in G_4 depends on b, yet for b to be defended, q must already have been established; this pattern does not exist in G_1, where q does not depend on b. ⌟

In general, there may be more than one grounded argument graph for a given framework, as the following example shows.

Example 10. Consider the ABA framework:

$$\mathcal{L} = \{\ p, q, r, x, a, b, c, d, e, f\ \}$$
$$\mathcal{R} = \{\ p \leftarrow a,\quad p \leftarrow b,\quad q \leftarrow c,\quad r \leftarrow d\ \}$$
$$\mathcal{A} = \{\ a, b, c, d, e, f\ \}$$
$$\bar{a} = x,\ \bar{b} = x,\ \bar{c} = p,\ \bar{d} = q,\ \bar{e} = f,\ \bar{f} = e$$

Fig. 13 shows all the complete argument graphs for this framework. These are also admissible (there are additional admissible argument graphs, e.g., the empty argument graph). It is evident that the two left-most graphs are $\subseteq$-minimal as well as grounded. ⌟

The following two theorems establish the relations between argument graph semantics and extension-based semantics.

Theorem 4.20. *Let G be an argument graph and A_G the set of arguments represented in G.*

i. *If G is admissible, so is A_G.*
ii. *If G is complete, then there is a complete extension A^* such that $\mathsf{A}_G \subseteq \mathsf{A}^*$ and $claims(\mathsf{A}_G) = claims(\mathsf{A}^*)$.*
iii. *If G is grounded and A^* is the grounded extension, then $\mathsf{A}_G \subseteq \mathsf{A}^*$ and $claims(\mathsf{A}_G) = claims(\mathsf{A}^*)$.* ⌟

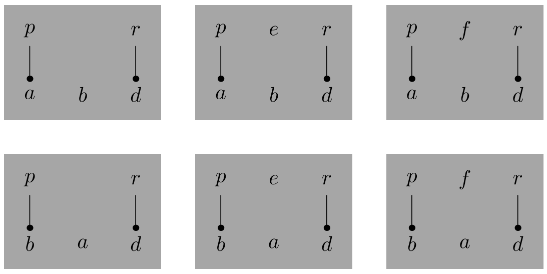


Fig. 13. Complete argument graphs for Example 10.

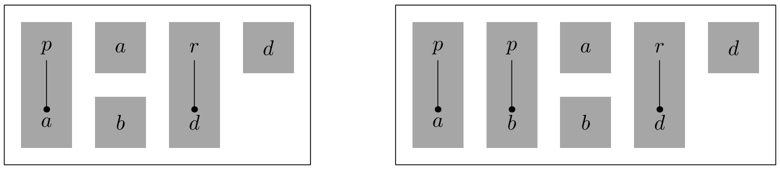


Fig. 14. To the left: the arguments represented by the top-left argument graph from Fig. 13. To the right is the grounded extension. (Both concern the framework of Example 10.)

As an illustration of this result, consider the top-left, grounded argument graph in Fig. 13 for the ABA framework in Example 10. The set of arguments it represents is shown to the left of Fig. 14, the grounded extension is shown to the right. As Theorem 4.20 ensures, the left-hand set is a subset of the right-hand set.

The converse direction of Theorem 4.20 holds, making use of the notion of *graphical conversion* from Definition 4.5, as follows:

Theorem 4.21. *Let* A *be a set of tree-based arguments.*

i. *If* A *is admissible, then for all conversions* G *of* A, G *is admissible.*
ii. *If* A *is complete, then for all conversions* G *of* A, G *is complete.*
iii. *If* A *is grounded, then there exists a conversion* G *of* A *such that* G *is grounded.* ⌟

The existential quantifier of part (iii), here, cannot be strengthened to a universal, as Example 9 shows. For let A be the set of arguments that are either represented in G_1 or in G_4; A is the grounded extension. Both G_1 and G_4 are conversions of A. Since G_1 is grounded, then (iii) is confirmed, but the fact that G_4 is not grounded shows that we cannot strengthen the existential quantifier. Note that, as a corollary of this result and existing results on existence of grounded and admissible extensions for all ABA frameworks [6,11], grounded and admissible argument graphs are also always guaranteed to exist, for any ABA framework.

Recall now the notion of a sentence s being admissible or grounded with respect to some $A \subseteq \mathcal{A}$ (defined in §2). The following corollary, which follows easily from ones already proven, shows that this notion is matched by a corresponding notion defined in terms of argument graphs.

Corollary 4.22. *(i) Let* A *be an admissible (respectively grounded) extension, with* a $\in$ A *such that* $claim(\mathsf{a}) = s$*. Then there is an admissible (respectively grounded) argument graph* G *such that* $s \in v(G)$*. (ii) Let* G *be an admissible (respectively grounded) argument graph with* $s \in v(G)$*. Then there is an admissible (respectively grounded) extension* A *and some* a $\in$ A *such that* $s = claim(\mathsf{a})$.

Proof. Part (i) is an easy consequence of Theorem 4.21. Part (ii) is an easy consequence of Theorem 4.20. □

We will use this last result in the following section, in relation to the soundness and completeness of the dispute derivations we therein define.

5. Graphical dispute derivations

In this section we define the novel computational machinery of *graphical dispute derivations* (graph-DDs in short) for determining whether a given sentence is supported by an admissible or grounded argument graph, and computing an admissible argument graph for showing that this is the case. These graph-DDs can be seen as an evolution of the dispute derivations of [12,14,27] for ABA and of [26] for abstract argumentation. Like their predecessors, graph-DDs are finite sequences of tuples, each tuple representing a state of play in a game between two fictional players, PROPONENT and OPPONENT.

However, whereas their ABA predecessors use sets of assumptions or intermediate steps in the construction of arguments to represent a state of play, graph-DDs use intermediate steps in the construction of argument graphs. Moreover, graph-DDs use a graph to guarantee termination and completeness in the grounded case, inspired by [26], but having sentences rather than arguments as nodes. Like the X-dispute derivations of [27], graph-DDs are defined parametrically, with specific choices of parameters supporting computation under different semantics; however, whereas graph-DDs are defined for admissible and grounded argument graphs, X-dispute derivations are defined for admissible, grounded and ideal sets of assumptions/arguments.

Before we formally define graph-DDs in §5.2, we give some preliminary definitions in §5.1. We illustrate graph-DDs in §5.3, also pointing out differences with the dispute derivations of [12,14,27]. We prove soundness and completeness of graph-DDs with respect to the semantics of admissible and grounded argument graphs in §5.4, and discuss in §5.5 an implementation of graph-DDs we have used for experimentation in §6.

5.1. Preliminaries

In graph-DDs, argument graphs are built gradually, starting with a single sentence. This process requires some means of marking which sentences s of a growing argument graph have already been processed—where processing typically involves, if $s \notin \mathcal{A}$, extending the graph at s by adding edges to other sentences in a rule whose head is s. *Potential argument graphs* are intermediate steps in the construction of argument graphs where sentences are marked or unmarked, defined as follows.

Definition 5.1. A *potential argument graph* G is a directed acyclic graph equipped with a set of *unmarked* sentences $u(G) \subseteq sinks(G)$, where $v(G) \subseteq \mathcal{L}$ and for all $s \in v(G)$:

i. if $s \in \mathcal{A}$, then $s \in sinks(G)$;
ii. if $s \notin \mathcal{A}$, then either (a) $s \in u(G)$; or (b) $s \notin u(G)$ and there is a rule $(s \leftarrow s_1, \ldots, s_n) \in \mathcal{R}$ such that there is an edge (s, s') in $e(G)$ iff $s' \in \{s_1, \ldots, s_n\}$.

If $s \in u(G)$ it is said to be *unmarked*; if $s \in v(G) \setminus u(G)$ it is said to be *marked*. We sometimes write $m(G)$ for $v(G) \setminus u(G)$.

G is said to be *focused* if it has a unique source, called the *claim* of G and represented as *claim*(G). The *support* of G, written *support*(G), is $v(G) \cap \mathcal{A}$.

If $u(G) = \emptyset$ then G is also referred to as an *actual argument graph*. ⌟

Note that, if we ignore the (empty set of) unmarked sentences, an actual argument graph G is an argument graph according to Definition 4.1. We call this the *corresponding argument graph* of G. Note also that we sometimes elide the 'potential' qualification where this causes no ambiguity.

An actual argument graph corresponds to the situation where all of the beliefs of an agent which stand in need of inferential support have an appropriate support—if one is needed—with respect to the beliefs of that agent, and this is known to the agent. The existence of an appropriate support is secured for non-assumptions by condition (ii)(b) in Definition 5.1; assumptions need no support, and indeed cannot have one—this is ensured by condition (i). The knowledge to the agent is represented by the condition on actuality of the argument, that $u(G) = \emptyset$. Potential argument graphs, by contrast, correspond to the situation where some of the beliefs of an agent which require support do not yet have it, *or* to the situation where an agent has beliefs that do not require inferential support, but the agent does not yet know this.

Example 11. Consider the following ABA framework:

$$\begin{aligned}
&\mathcal{L} = \{\, p, q, r, s, t, x, y, a, b \,\}\\
&\mathcal{A} = \{\, a, b \,\}\\
&\mathcal{R} = \{\, p \leftarrow q, r,\\
&\qquad q \leftarrow a,\\
&\qquad r \leftarrow t, a, b,\\
&\qquad s \leftarrow b,\\
&\qquad t \leftarrow \,\}\\
&\bar{a} = x,\ \bar{b} = y
\end{aligned}$$

Consider graphs G_1 and G_2, shown to the left and right, respectively, in Fig. 15. Now, G_1 is necessarily merely a potential argument graph, since it must be that $s, r \in u(G_1)$. (That $a \in u(G_1)$ and that $a \notin u(G_1)$ are both consistent with Definition 5.1.) However, G_2 may be either potential or actual: for any of a, t, or b—or none of them—might be members of $u(G_2)$. If G_2 is potential and $t \in u(G_2)$ then t is a belief that requires no inferential support but the agent does not yet know this. ⌟

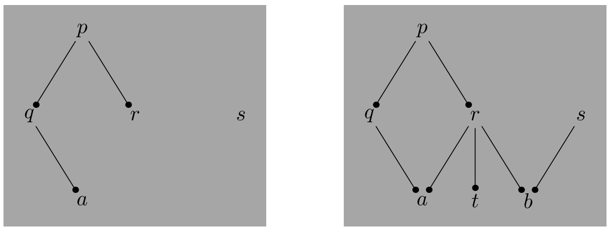


Fig. 15. Graph G_1 (left) and graph G_2 (right) for Example 11.

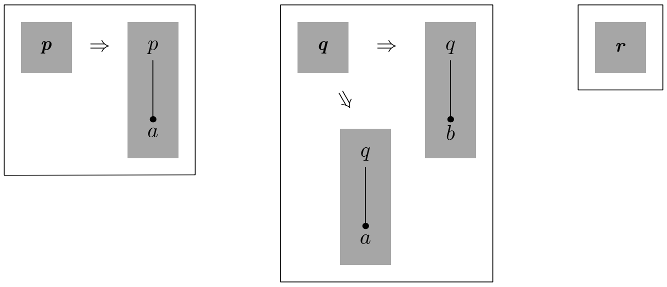


Fig. 16. Potential argument graphs for Example 12: in the leftmost box, G_p, and $G_{p,a}$; in the middle box, G_q, $G_{q,a}$ and $G_{q,b}$; in the rightmost box, G_r. The unmarked sentences are given in bold (the sentences $\boldsymbol{p}$, $\boldsymbol{q}$ and $\boldsymbol{r}$ from G_p, G_q and G_r, respectively).

It may be possible to expand a potential argument graph to form a single actual argument graph, several actual argument graphs, or not to expand it further at all, as illustrated by the following example.

Example 12. Consider the ABA framework shown below.

$$\begin{aligned}
&\mathcal{L} = \{\, p, q, r, z, a, b \,\}\\
&\mathcal{A} = \{\, a, b \,\}\\
&\mathcal{R} = \{\, p \leftarrow a,\\
&\qquad q \leftarrow a,\\
&\qquad q \leftarrow b \,\}\\
&\bar{a} = z,\ \bar{b} = z
\end{aligned}$$

Consider the potential argument graphs shown in Fig. 16. The argument graph G_p, shown on the left of the leftmost box, is such that $v(G_p) = u(G_p) = \{p\}$, and $e(G_p) = \emptyset$: this is a potential argument graph in which p is unmarked. G_p can be expanded into the actual argument graph $G_{p,a}$ shown on the right of the leftmost box, where $u(G_{p,a}) = \{a\}$ or $u(G_{p,a}) = \emptyset$; this corresponds to an argument graph as given by Definition 4.1, if we ignore the marking. Given the absence of any rule in $\mathcal{R}$ other than $p \leftarrow a$, there is no other actual (or indeed, potential) argument graph than $G_{p,a}$ which is an expansion of G_p.

By way of contrast, the leftmost potential argument graph in the middle box, G_q, with $v(G_q) = u(G_q) = \{q\}$, can be expanded into both $G_{q,a}$ and $G_{q,b}$, as shown in Fig. 16; this is allowed by the presence of both $q \leftarrow a$ and $q \leftarrow b$ in the set of rules.

Finally, note that G_r with $v(G_r) = u(G_r) = \{r\}$, in the rightmost box, cannot be expanded into an actual argument graph, since there is no rule in $\mathcal{R}$ whose head is r. ⌟

The notion of expansion used informally in Example 12 can be formalized as follows.

Definition 5.2. Let G be a potential argument graph. An *expansion of* G is a potential argument graph G' such that:

- $v(G) \subseteq v(G')$;
- $e(G) \subseteq e(G')$;
- $m(G) \cap u(G') = \emptyset$;
- if $s \in v(G)$ and $\{s' \mid (s, s') \in e(G)\}$ is non-empty, then $\{s' \mid (s, s') \in e(G)\} = \{s' \mid (s, s') \in e(G')\}$. ⌟

(Note that this allows a potential argument graph to be an expansion of itself.)

Table 1
Definition of $\cup_g$.

	$\mathcal{G} \cup_g E$
$X = \text{ADM}$	$\mathcal{G}$
$X = \text{GRN}$	$(v(\mathcal{G}) \cup \{x \mid (s, s') \in E, (x = s \vee x = s')\}, e(\mathcal{G}) \cup E)$

Thus, an expansion G' of a potential argument graph G is another potential argument graph which 'grows' G' in a particular way, by choosing rules to justify all members of $u(G)$. We saw this in Example 12, where G_p was expanded into $G_{p,a}$ by the rule $p \leftarrow a$; and G_q may be grown into $G_{q,a}$ or $G_{q,b}$ depending on whether the rule $q \leftarrow a$ or $q \leftarrow b$ is used. Note that, trivially, the expansion of a *focused* potential argument graph is a *focused* argument graph with the same claim.

Our graph-DDs manipulate *sets* of potential argument graphs (generated by the OPPONENT player), which are equipped with machinery for distinguishing their marked (i.e., processed) and unmarked members. We refer to these sets as *argument graph sets*, defined as follows.

Definition 5.3. An *argument graph set* $\boldsymbol{O}$ is a set of potential argument graphs, equipped with a set $u(\boldsymbol{O}) \subseteq \boldsymbol{O}$ of the *unmarked* members. If $G \in u(\boldsymbol{O})$ we say that G is *unmarked* in $\boldsymbol{O}$; if $G \in \boldsymbol{O} \setminus u(\boldsymbol{O})$ we say that G is *marked* in $\boldsymbol{O}$. ⌟

Note that this definition implies that, if $\boldsymbol{O}$ is an argument graph set and $\boldsymbol{O} = \emptyset$, then $u(\boldsymbol{O}) = \emptyset$.

To simplify the presentation of graph-DDs, it will prove convenient to define the following operations in relation to potential argument graphs and argument graph sets. Some are overloaded.

Definition 5.4. In the following, G and G' are potential argument graphs; $s \in \mathcal{L}$; $S \subseteq \mathcal{L}$; $\boldsymbol{O}$ and $\boldsymbol{O}'$ are argument graph sets; and X is a set of potential argument graphs (*not* an argument graph set).

- *newgrph*(s) is G, where $v(G) = u(G) = \{s\}$ and $e(G) = \emptyset$.
- $G \cup_u S$ is G', where $v(G') = v(G) \cup S$, $e(G') = e(G)$ and $u(G') = u(G) \cup S$.
- $G \cup_m S$ is G', where $v(G') = v(G) \cup S$, $e(G') = e(G)$ and $u(G') = u(G) \setminus S$.
- $\boldsymbol{O} \cup_u X$ is $\boldsymbol{O}'$, where $\boldsymbol{O}'$ contains just the argument graphs in $\boldsymbol{O}$ and X, and $u(\boldsymbol{O}') = u(\boldsymbol{O}) \cup X$.
- $\boldsymbol{O} \cup_m X$ is $\boldsymbol{O}'$, where $\boldsymbol{O}'$ contains just the argument graphs in $\boldsymbol{O}$ and X, and $u(\boldsymbol{O}') = u(\boldsymbol{O}) \setminus X$.
- $\boldsymbol{O} \setminus X$ is $\boldsymbol{O}'$, where $\boldsymbol{O}'$ contains just the argument graphs in $\boldsymbol{O}$ but not in X, and $u(\boldsymbol{O}') = u(\boldsymbol{O}) \setminus X$.
- *updtgrph*$(G, s \leftarrow s_1, \ldots, s_n, S)$, where $(s \leftarrow s_1, \ldots, s_n) \in \mathcal{R}$, is G', where:
  - $v(G') = v(G) \cup \{s_1, \ldots, s_n\}$;
  - $e(G') = e(G) \cup \{(s, s') \mid s' \in \{s_1, \ldots, s_n\}\}$;
  - $u(G') = (u(G) \cup \{s' \in \{s_1, \ldots, s_n\} \mid s' \notin m(G)\}) \setminus (\{s\} \cup S)$. ⌟

In words, these operations are as follows. *newgrph* takes a sentence $s \in \mathcal{L}$ and forms a new potential argument graph in which s is unmarked. $G \cup_u S$ is the result of adding the sentences in S unmarked to the argument graph G, and leaving G otherwise unchanged; $G \cup_m S$ is the result of adding the sentences in S marked to G. $\boldsymbol{O} \cup_u X$ adds the argument graphs in X unmarked to $\boldsymbol{O}$, and $\boldsymbol{O} \cup_m X$ adds the argument graphs in X marked to $\boldsymbol{O}$. $\boldsymbol{O} \setminus X$ removes all argument graphs in X from $\boldsymbol{O}$. Finally, *updtgrph*$(G, s \leftarrow s_1, \ldots, s_n, S)$ extends G at s by adding $s_1, \ldots, s_n$ to G's vertices (where not already present), and by adding edges from s to each of $s_1, \ldots, s_n$; any s_i which was not already marked in G is, in *updtgrph*$(G, s \leftarrow s_1, \ldots, s_n, S)$, unmarked—unless s_i is in S.

5.2. Graphical dispute sequences and derivations

In this section we give the full definition of the graph-DDs used to determine whether some sentence $s \in \mathcal{L}$ is admissible/grounded. The basic concept is that of an *X-graphical dispute sequence* (graph-DS in short), where X can be either ADM (for 'admissible') or GRN (for 'grounded'). Successful graph-DDs are graph-DSs of a particular form.

In defining graph-DSs, we make use of a concept of *selection*, in the same spirit of [12,14,27]. Given the ith tuple, this chooses, for some specified component in the tuple, an element of that component. This element is then operated on, to form the $(i+1)$th tuple. Informally, the ith tuple in a graph-DS consists of

- a PROPONENT potential argument graph $\mathcal{P}_i$;
- an OPPONENT argument graph set $\boldsymbol{O}_i$;
- a graph $\mathcal{G}_i$ whose nodes are sentences;
- a set of assumptions D_i (the PROPONENT *defences*);
- a set of assumptions C_i (the OPPONENT *culprits*).

Moreover, the definition of graph-DSs uses the operator $\cup_g$ defined in Table 1. $\cup_g$ is defined parametrically according to whether X is ADM or GRN, for $\mathcal{G}$ a graph and E a set of pairs of sentences (edges). Thus, substantially, $\mathcal{G} \cup_g E$ is only playing a role when $X = \textsc{grn}$ (as it ignores E for $X = \textsc{adm}$). Finally, in the definition of graph-DSs, for any graph $\mathcal{G}$ or PROPONENT potential argument graph $\mathcal{P}$, $acyclic(\mathcal{G})/acyclic(\mathcal{P})$ is true iff $\mathcal{G}/\mathcal{P}$ (respectively) is acyclic.

In presenting the definition we follow the convention that where elements of a tuple do not change from the ith to the $(i+1)$th step, then this is not noted explicitly, and we typically write rules in $\mathcal{R}$ in the form $s \leftarrow R$ where R is the set of sentences in the body of the rule.

Definition 5.5. Let $s_0 \in \mathcal{L}$. Let n be such that $0 \leqslant n \leqslant \omega$.[6] An *X-graphical dispute sequence* (X-graph-DS, for $X \in \{\textsc{adm}, \textsc{grn}\}$) *for* s_0 *of length* n is a sequence $((\mathcal{P}_i, \boldsymbol{O}_i, \mathcal{G}_i, D_i, C_i))_{i=0}^{n}$, where:

$$\mathcal{P}_0 = newgrph(s_0)$$

$$\boldsymbol{O}_0 = \emptyset$$

$$\mathcal{G}_0 = \begin{cases} \lozenge & \text{if } X = \textsc{adm} \\ (\{s_0\}, \emptyset) & \text{if } X = \textsc{grn} \end{cases}$$

$$D_0 = \mathcal{A} \cap \{s_0\}$$

$$C_0 = \emptyset$$

and for every i such that $0 \leqslant i < n$, only one $s \in u(\mathcal{P}_i)$ or one $G \in u(\boldsymbol{O}_i)$ is *selected* and

1. if $s \in u(\mathcal{P}_i)$ is *selected*, then
   (i) if $s \in \mathcal{A}$, then:

$$\mathcal{P}_{i+1} = \mathcal{P}_i \cup_m \{s\}$$

$$\boldsymbol{O}_{i+1} = \begin{cases} \boldsymbol{O}_i & \text{if } \exists G \in \boldsymbol{O}_i \text{ such that } \bar{s} = claim(G) \\ \boldsymbol{O}_i \cup_u \{newgrph(\bar{s})\} & \text{otherwise} \end{cases}$$

$$\mathcal{G}_{i+1} = \mathcal{G}_i \cup_g \{(\bar{s}, s)\}$$

   and $acyclic(\mathcal{G}_{i+1})$;
   (ii) if $s \notin \mathcal{A}$, then there is some $(s \leftarrow R) \in \mathcal{R}$ such that $R \cap C_i = \emptyset$, and

$$\mathcal{P}_{i+1} = updtgrph(\mathcal{P}_i, s \leftarrow R, \emptyset)$$

$$\mathcal{G}_{i+1} = \mathcal{G}_i \cup_g \{(s', s) \mid s' \in R\}$$

$$D_{i+1} = D_i \cup (R \cap \mathcal{A})$$

   and $acyclic(\mathcal{P}_{i+1})$, $acyclic(\mathcal{G}_{i+1})$;
2. if $G \in u(\boldsymbol{O}_i)$ and $s \in u(G)$ are *selected*, then
   (i) if $s \in \mathcal{A}$, then:
   (a) either s is ignored, i.e.:

$$\boldsymbol{O}_{i+1} = (\boldsymbol{O}_i \setminus \{G\}) \cup_u \{G \cup_m \{s\}\}$$

   (b) or $s \notin D_i$ and $s \in C_i$, and:

$$\boldsymbol{O}_{i+1} = (\boldsymbol{O}_i \setminus \{G\}) \cup_m \{G \cup_m \{s\}\}$$

$$\mathcal{G}_{i+1} = \mathcal{G}_i \cup_g \{(\bar{s}, claim(G))\}$$

   and $acyclic(\mathcal{G}_{i+1})$;
   (c) or $s \notin D_i$ and $s \notin C_i$, and:

$$\mathcal{P}_{i+1} = \begin{cases} \mathcal{P}_i & \text{if } \bar{s} \in v(\mathcal{P}_i) \\ \mathcal{P}_i \cup_u \{\bar{s}\} & \text{otherwise} \end{cases}$$

$$\boldsymbol{O}_{i+1} = (\boldsymbol{O}_i \setminus \{G\}) \cup_m \{G \cup_m \{s\}\}$$

$$\mathcal{G}_{i+1} = \mathcal{G}_i \cup_g \{(\bar{s}, claim(G))\}$$

$$D_{i+1} = D_i \cup (\{\bar{s}\} \cap \mathcal{A})$$

$$C_{i+1} = C_i \cup \{s\}$$

   and $acyclic(\mathcal{G}_{i+1})$;

[6] As conventionally, ω is the least infinite ordinal.

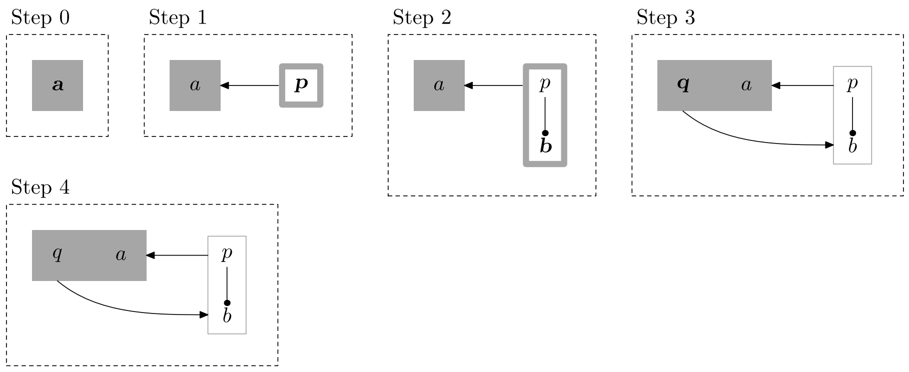


Fig. 17. A depiction of an ADM-graph-DS for a for Example 13.

Table 2
Tabular ADM-graph-DS for the same derivation shown in Fig. 17.

Step	Case	$\mathcal{P}_i$	$\mathbf{O}_i$ (only $u(\mathbf{O}_i)$ shown)	D_i	C_i
0	n/a	$(\{\mathbf{a}\}, \emptyset)$	$\emptyset$	$\{a\}$	$\emptyset$
1	1(i)	$(\{a\}, \emptyset)$	$\{(\{\mathbf{p}\}, \emptyset)\}$	$\{a\}$	$\emptyset$
2	2(ii)	$(\{a\}, \emptyset)$	$\{(\{p, \mathbf{b}\}, \{(p, b)\})\}$	$\{a\}$	$\emptyset$
3	2(i)(c)	$(\{\mathbf{q}, a\}, \emptyset)$	$\emptyset$	$\{a\}$	$\{b\}$
4	1(ii)	$(\{q, a\}, \emptyset)$	$\emptyset$	$\{a\}$	$\{b\}$

(ii) if $s \notin \mathcal{A}$, let:

$$\mathsf{R}_C = \begin{cases} \emptyset & \text{if } X = \textsc{grn} \\ \{R \mid (s \leftarrow R) \in \mathcal{R},\ R \cap C_i \neq \emptyset,\ acyclic(updtgrph(G, s \leftarrow R, \emptyset))\} & \text{otherwise} \end{cases}$$

$$\mathsf{R}_{\neg C} = \{R \mid (s \leftarrow R) \in \mathcal{R},\ acyclic(updtgrph(G, s \leftarrow R, \emptyset)),\ R \notin \mathsf{R}_C\};$$

then:

$$\begin{aligned}\mathbf{O}_{i+1} = {} & ((\mathbf{O}_i \setminus \{G\}) \cup_m \{updtgrph(G, s \leftarrow R, C_i) \mid R \in \mathsf{R}_C\}) \\ & \cup_u \{updtgrph(G, s \leftarrow R, \emptyset) \mid R \in \mathsf{R}_{\neg C}\}. \quad \lrcorner\end{aligned}$$

Several examples of (finite) graph-DSs are given in §5.3, to illustrate different aspects of the sequences. The following is a simple example.

Example 13. Consider the ABA framework:

$$\begin{aligned} \mathcal{L} &= \{\, p, q, a, b \,\} \\ \mathcal{A} &= \{\, a, b \,\} \\ \mathcal{R} &= \{\, p \leftarrow b, \\ &\qquad q \leftarrow \,\} \\ \bar{a} &= p,\ \bar{b} = q \end{aligned}$$

A sample ADM-graph-DS of length 4 for a is represented in Fig. 17. Here and in the remainder, (a representation of the components of the tuple in) each step is enclosed within dashed lines. The PROPONENT argument graphs are shaded, and the OPPONENT argument graphs are in solid lines (OPPONENT graphs are in thick solid lines if unmarked, in thin lines if marked). Within an argument graph, the unmarked sentences are depicted in bold. The same sequence is represented in tabular form in Table 2 (with the same convention on unmarked sentences as in Fig. 17). Here, we also show the case of the definition of an ADM-graph-DS which was applied (cf. Definition 5.5), and we only show the unmarked opponent argument graphs. Moreover, we omit (in both presentations of the sequence) the $\mathcal{G}_i$ component (as this is always the empty graph for an ADM-graph-DS).

Note that this sequence is also a GRN-graph-DS, if the components $\mathcal{G}_i$ are added as in Table 3. The acyclicity check is clearly passed by $\mathcal{G}_i$ at each step.

Note further that any initial sequence of the graph-DS here is also a graph-DS; this is true in general, both for ADM-graph-DSs and GRN-graph-DSs. $\lrcorner$

Table 3
$\mathcal{G}_i$ components for GRN-graph-DS version of the ADM-graph-DS in Fig. 17 and Table 2.

Step	$\mathcal{G}_i$
0	$(\{a\}, \emptyset)$
1	$(\{a, p\}, \{(p, a)\})$
2	$(\{a, p\}, \{(p, a)\})$
3	$(\{a, p, q\}, \{(p, a), (q, p)\})$
4	$(\{a, p, q\}, \{(p, a), (q, p)\})$

Some commentary on Definition 5.5 is appropriate here; we take the cases by turn.

1. A sentence s is selected amongst the unmarked sentences in the PROPONENT potential argument graph, for attempted expansion of this graph.
   1(i). If this selected sentence s is an assumption, then where this has already been considered as a point of attack (as determined by seeing that there is an OPPONENT focused potential argument graph G in the current argument graph set $\boldsymbol{O}_i$ with claim the contrary of the sentence), no further processing needs to be performed on s. Otherwise, the existence of OPPONENT focused argument graphs attacking s needs to be determined and each such graph needs to be counter-attacked, so a new OPPONENT focused potential argument graph with just $\bar{s}$ as a node is added to the argument graph set (to form $\boldsymbol{O}_{i+1}$). In both cases the selected assumption is marked.
   1(ii). If s is not an assumption, then the PROPONENT potential argument graph is expanded by adding edges corresponding to sentences in the body of a chosen rule whose head is s, and marking s in the resulting potential argument graph. The rule chosen for expansion needs to have no current culprits amongst its assumptions ($R \cap C_i = \emptyset$). The fact that s was unmarked ensures that it has no existing children in the PROPONENT potential argument graph. An acyclicity check is performed on the resulting PROPONENT potential argument graph (to ensure the result of the update is *still* a potential argument graph). Every assumption in R is added to D_i, to remember that it is necessarily a defence for the PROPONENT.

   After each of these cases, if $X = \text{GRN}$, $\mathcal{G}_i$ will have been updated, and an acyclicity check performed.
2. An OPPONENT potential argument graph (in the current OPPONENT argument graph set $\boldsymbol{O}_i$) is selected for expansion or for being counter-attacked, and an unmarked s selected from the graph.
   2(i)(a). The selected sentence s is an assumption, which is ignored. The graph-DS would then have to find another means of attacking the selected argument graph, which is therefore returned unmarked to $\boldsymbol{O}_{i+1}$, but with s marked.
   2(i)(b). The selected sentence s is an assumption which is already attacked by the PROPONENT (since it belongs to the current set of culprits). In this case the OPPONENT argument graph can be marked (with s also marked).
   2(i)(c). s is an assumption which is not yet known to be attacked, and is not amongst the PROPONENT defences D_i. If $\bar{s}$ is in the PROPONENT potential argument graph, then there is in fact an attack against s already, so that no further changes to the PROPONENT graph are necessary. If $\bar{s}$ is not currently in the PROPONENT potential argument graph, then it is added, so that at some later stage the PROPONENT must find a way to argue for it. In both cases the OPPONENT argument graph can also be marked (with the selected sentence also marked). Moreover, the selected sentence is remembered as a culprit (in C_{i+1}) and its contrary, if an assumption, is remembered as a defence (in D_{i+1}).

   After each of these cases, if $X = \text{GRN}$ and $\mathcal{G}_i$ has been updated (cases 2(i)(b) and 2(i)(c)), then an acyclicity check on it is performed.
   2(ii). If s is not an assumption, then the OPPONENT argument graph from which it was selected can be expanded in as many ways as the number of rules with s as their head. For $X = \text{ADM}$ things work as follows. Some of those rules (R_C) will produce expansions which are already attacked (as they have a current culprit in their body); these are added marked to $\boldsymbol{O}_i$ to form $\boldsymbol{O}_{i+1}$. The others ($\mathsf{R}_{\neg C}$) produce expansions which are not already attacked; these expansions must be added as unmarked to form $\boldsymbol{O}_{i+1}$, so that the PROPONENT must find counter-attacks at a later stage.
   In the case of $X = \text{GRN}$, all expansions are considered as not currently attacked, independently of the presence of existing culprits in their bodies.
   In both cases ($X = \text{GRN}$ and $X = \text{ADM}$), rules introducing cycles into the selected OPPONENT potential argument graph G are ignored, as, intuitively, they are not possibly contributing to generating attacks.
   In the case of those expanded argument graphs which are added unmarked to form $\boldsymbol{O}_{i+1}$, then all new sentences are unmarked; for the expanded argument graphs which are added marked to form $\boldsymbol{O}_{i+1}$, those sentences are unmarked which are new and not members of C_i, i.e., not culprits already.

Note that in general we perform an acyclicity check on potential argument graphs as they are being constructed. This occurs, for the PROPONENT, in case 1(ii) of Definition 5.5, where $\mathcal{P}_i$ is expanded; and in case 2(ii) for the OPPONENT, where a member of $\boldsymbol{O}_i$ is expanded (possibly, in many different ways). Since argument graphs must be acyclic by definition, this ensures that the graphs we build conform to that definition, while at the same time guaranteeing termination. Overall, the following theorem establishes the coherence of Definition 5.5.

Theorem 5.6. *Let* $((\mathcal{P}_i, \boldsymbol{O}_i, \mathcal{G}_i, D_i, C_i))_{i=0}^{n}$ *be an X-graph-DS for* s_0*. Then for all i such that* $0 \leqslant i \leqslant n$ *(if n is finite), or all i such that* $0 \leqslant i < n$ *(otherwise—i.e., if* $n = \omega$*):*

i. $\mathcal{P}_i$ *is a potential argument graph, and* $s_0 \in v(\mathcal{P}_i)$*;*
ii. $\boldsymbol{O}_i$ *is an argument graph set;*
iii. $\mathcal{G}_i$ *is a directed graph over* $\mathcal{L}$*;*
iv. $D_i \subseteq \mathcal{A}$ *and* $C_i \subseteq \mathcal{A}$*.* ⌟

A graph-DD is a finite graph-DS with a particular constraint on the last tuple, as follows.

Definition 5.7. Let $s_0 \in \mathcal{L}$. An *X-graphical dispute derivation* (X-graph-DD, $X \in \{\textsc{adm}, \textsc{grn}\}$) *for* s_0 *with resulting (potential argument graph)* $\mathcal{P}_n$ is an X-graph-DS for s_0 of length $n < \omega$:

$$(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$$

where $u(\mathcal{P}_n) = \emptyset$ and $u(\boldsymbol{O}_n) = \emptyset$. ⌟

The graph-DS in Example 13 is a graph-DD. The constraints in Definition 5.7 mean that a graph-DD must terminate with no members of $v(\mathcal{P}_n)$ being without a justification ($u(\mathcal{P}_n) = \emptyset$), and all OPPONENT potential argument graphs in $\boldsymbol{O}_n$ having been processed ($u(\boldsymbol{O}_n) = \emptyset$). The first constraint amounts to imposing that the resulting potential argument graph of a graph-DD is in fact an actual argument graph, as sanctioned by the following.

Corollary 5.8. *Let* $X \in \{\textsc{adm}, \textsc{grn}\}$ *and* $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$ *be an X-graph-DD for* s_0*. Then* $\mathcal{P}_n$ *is an actual argument graph.*

Proof. By Theorem 5.6, $\mathcal{P}_n$ is a potential argument graph. Definition 5.7 requires that $u(\mathcal{P}_n) = \emptyset$, so by Definition 5.1, $\mathcal{P}_n$ is actual. □

By virtue of this result, we often use the argument graph $\mathcal{P}$ corresponding to $\mathcal{P}_n$ as resulting from an X-graph-DD $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$.

Given an X-graph-DS $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_i, \boldsymbol{O}_i, \mathcal{G}_i, D_i, C_i)$, there are several indeterminacies in Definition 5.5, which give the X-graph-DS the possibility of being continued in different ways to give an X-graph-DD. The indeterminacies fall into two categories: (A) those that make no difference to the overall outcome, i.e. to whether the X-graph-DS can be extended into an X-graph-DD and which X-graph-DDs are possible; (B) those which do make such a difference. Into the first category fall:

A1. the selection of either $s \in u(\mathcal{P}_i)$ or $G \in u(\boldsymbol{O}_i)$ and $s \in u(G)$—determining whether case (1) or case (2) in Definition 5.5 applies;
A2. given a choice between cases (1) and (2) of Definition 5.5, the selection of the particular members of the sets—i.e. for case (1) *which* $s \in u(\mathcal{P}_i)$ is selected; for case (2), *which* $G \in u(\boldsymbol{O}_i)$ and $s \in u(G)$ are selected.

That these indeterminacies make no difference to the overall outcome can be seen intuitively. For A1, it is evident that all $s \in u(\mathcal{P}_i)$ and $G \in u(\boldsymbol{O}_i)$ must be considered at some point, since for there to be a graph-DD, each of these sets must eventually be empty. For A2, again, it is plain that, in the PROPONENT case (1), all $s \in u(\mathcal{P}_i)$ must eventually be selected and treated according to Definition 5.5—intuitively, all $s \in u(\mathcal{P}_i)$ are pending proof, and if the potential argument graph is to become an actual argument graph, everything the PROPONENT needs to support or defend must be proved. In the OPPONENT case, the reasoning is similar: each $G \in u(\boldsymbol{O}_i)$ represents a possible challenge on the part of the OPPONENT to some assumption the PROPONENT has made use of; and all such challenges must be met. Also, within a specific OPPONENT G, if s is selected which does *not* lead to a successful counter-attack by the PROPONENT, then this can be dealt with by the 'ignore' case, which *is* relevant to eventual success of the graph-DD (see B2).

Evidently, it is not always the case that a selection can be made: if $\mathcal{P}_i$ is empty but $\boldsymbol{O}_i$ is not, then the X-graph-DS can only be continued according to case (2) in Definition 5.5. The selection of type (A1) can actually be equated to the selection of a *player* amongst PROPONENT and OPPONENT. This selection can be made explicit in the specification of graph-DSs and graph-DDs, similarly [27], by means of a function *player* : $\mathbb{N} \rightarrow \{\textsc{proponent}, \textsc{opponent}\}$ that can be constrained to select a player at any step in a graph-DD only if its component if non-empty.

The indeterminacies which *do* make a difference to the overall outcome are:

B1. the choice of a rule $s \leftarrow R$ in case 1(ii);
B2. the choice between cases 2(i)(a) and either 2(i)(b) or 2(i)(c).

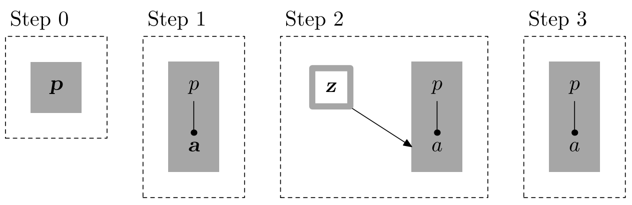


Fig. 18. ADM-graph-DD for p for Example 14.

Table 4
Tabulated form of the derivation from Fig. 18.

Step	Case	$\mathcal{P}_i$	$\boldsymbol{O}_i$ (only $u(\boldsymbol{O}_i)$ shown)	D_i	C_i
0	n/a	$(\{\boldsymbol{p}\}, \emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$
1	1(ii)	$(\{p, \boldsymbol{a}\}, \{(p,a)\})$	$\emptyset$	$\{a\}$	$\emptyset$
2	1(i)	$(\{p,a\}, \{(p,a)\})$	$\{(\{\boldsymbol{z}\}, \emptyset)\}$	$\{a\}$	$\emptyset$
3	2(ii)	$(\{p,a\}, \{(p,a)\})$	$\emptyset$	$\{a\}$	$\emptyset$

Indeed, several alternative rules may be possible in case 1(ii), and every assumption can be ignored if it does not belong already to the defence set (if it does, then case 2(i)(a) is the only option). We illustrate how B1 and B2 make a difference to the outcome of a graph-DD in §5.3.

The distinction between those indeterminacies which do, and those which do not, make a difference to which graph-DD a graph-DS may form part of, is relevant in implementations: for while parameters can be used to give strategies for making choices in each category, those of the latter need to be capable of being backtracked over in a search strategy. For example, if no rule for s exists in case (1)(ii) with no culprits in the body, then the graph-DS cannot be continued and backtracking needs to take place. We discuss this matter again in §5.5.

5.3. Examples of graph-DDs

Recall the indeterminacies of Definition 5.5, discussed at the end of Section 5.2. It is easy to see that B1 makes a difference to the result of a graph-DD, as follows.

Example 14. Consider the ABA framework:

$$\begin{aligned}
\mathcal{L} &= \{\, p, q, z, a \,\}\\
\mathcal{A} &= \{\, a \,\}\\
\mathcal{R} &= \{\, p \leftarrow q,\\
&\qquad p \leftarrow a \,\}\\
\bar{a} &= z
\end{aligned}$$

The graph G consisting of nodes $\{p, a\}$ and the single edge (p, a) is clearly both an admissible and a grounded argument graph. Fig. 18 depicts an ADM-graph-DD for p (which is also a GRN-graph-DD for p). Note that the movement from step 2 to step 3 in Fig. 18 removes the OPPONENT argument graph whose claim is z, since no rule can be used to prove z. The derivation is also shown as a sequence of tuples, in tabular form (and ignoring the $\mathcal{G}_i$ component), in Table 4. However, an ADM-graph-DS or GRN-graph-DS for p that picked the rule $p \leftarrow q$ to use at step 1(ii), after selecting p, could not be extended to a full ADM-graph-DD or GRN-graph-DD: indeed, informally, there is no way of proving q in the given ABA framework. Thus, B1 genuinely makes a difference to whether a graph-DS can be extended into a graph-DD. ⌟

That B2 makes a difference to the outcome of a derivation is shown in the following.

Example 15. Consider the ABA framework:

$$\begin{aligned}
\mathcal{L} &= \{\, p, q, z, r, a, b, c \,\}\\
\mathcal{A} &= \{\, a, b, c \,\}\\
\mathcal{R} &= \{\, p \leftarrow a,\\
&\qquad z \leftarrow b, c,\\
&\qquad q \leftarrow a,\\
&\qquad r \leftarrow \,\}
\end{aligned}$$

Table 5
GRN-graph-DD for p, for Example 15.

Step	Case	$\mathcal{P}_i$ / $\boldsymbol{O}_i$ (only $u(\boldsymbol{O}_i)$ shown)	$\mathcal{G}_i$	D_i / C_i
0	n/a	$(\{\boldsymbol{p}\}, \emptyset)$ $\emptyset$	$(\emptyset, \emptyset)$	$\emptyset$ $\emptyset$
1	1(ii)	$(\{p, \boldsymbol{a}\}, \{(p, a)\})$ $\emptyset$	$(\{p, a\}, \{(a, p)\})$	$\{a\}$ $\emptyset$
2	1(i)	$(\{p, a\}, \{(p, a)\})$ $\{(\{\boldsymbol{z}\}, \emptyset)\}$	$(\{p, a\}, \{(a, p), (z, a)\})$	$\{a\}$ $\emptyset$
3	2(ii)	$(\{p, a\}, \{(p, a)\})$ $\{(\{z, \boldsymbol{b}, \boldsymbol{c}\}, \{(z, b), (z, c)\})\}$	$(\{p, z, a, b\}, \{(a, p), (z, a)\})$	$\{a\}$ $\emptyset$
4	2(i)(a)	$(\{p, a\}, \{(p, a)\})$ $\{((\{z, b, \boldsymbol{c}\}, \{(z, b), (z, c)\})\}$	$(\{p, z, a, b\}, \{(a, p), (z, a)\})$	$\{a\}$ $\emptyset$
5	2(i)(c)	$(\{p, a, \boldsymbol{r}\}, \{(p, a)\})$ $\emptyset$	$(\{p, r, z, a, b\}, \{(a, p), (z, a), (r, z)\})$	$\{a\}$ $\{c\}$
6	1(ii)	$(\{p, a, r\}, \{(p, a)\})$ $\emptyset$	$(\{p, r, z, a, b\}, \{(a, p), (z, a), (r, z)\})$	$\{a\}$ $\{c\}$

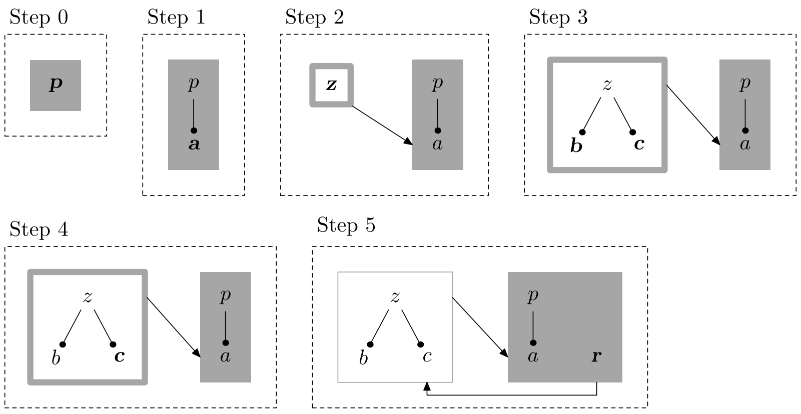


Fig. 19. Initial stages of the GRN-graph-DD for p, for Example 15.

$$\bar{a} = z \quad \bar{b} = q \quad \bar{c} = r$$

A GRN-graph-DD for p of length 6 is given in tabular form in Table 5 (its first five steps are also shown graphically in Fig. 19): (The final step, after r has been selected and proven using the rule $r \leftarrow$, is not shown in Fig. 19: this adds nothing to the graphical structures, since the rule used to establish r has an empty body.) This derivation illustrates the importance of case 2(i)(a) (the 'ignore' case) in Definition 5.5. In the move to step 4, case 2(i)(a) is applied, although b in the opponent argument graph is selected, it is ignored, with no proponent attack directed at b explored.

If case 2(i)(c) had been used instead at step 4, the corresponding sequence would have been as shown graphically in Fig. 20. This sequence is shown in tabular form in Table 6. The sequence of steps $(0, 1, 2, 3, 4^*, 5^*)$ here represent a GRN-graph-DS which cannot be extended to a GRN-graph-DD since $acyclic\mathcal{G}_{5^*}$ fails. This example is then also notable for the use of the $\mathcal{G}_i$ component in preventing infinite loops. An implementation would terminate at step 5* and realize that it must backtrack, ignore b and try to attack the OPPONENT argument on c instead (so as to give the graph-DD in Fig. 19).

Finally, this example shows an important difference with successful GB-dispute derivations of Dung et al. [14] (cf. Definition A.1 of Toni [27]). Consider the following attempt at a successful GB-dispute derivation for p for the ABA framework in the example shown in Table 7. Here, if the analogue of case 2(i)(c) is given priority over the 'ignore' case 2(i)(a), then implementations will loop infinitely, repeating steps 4–7. Our use of an acyclicity check on $\mathcal{G}_i$ prevents such infinite loops. ⌟

The next example further illustrates how graph-DDs relate to the dispute derivations of [12,14,27] and the advantages of our manipulation of (potential) argument graphs rather than (potential) argument trees in the original dispute derivations.

Example 16. Consider the following ABA framework.

$$\mathcal{L} = \{\ p, q, r, a, b\ \}$$

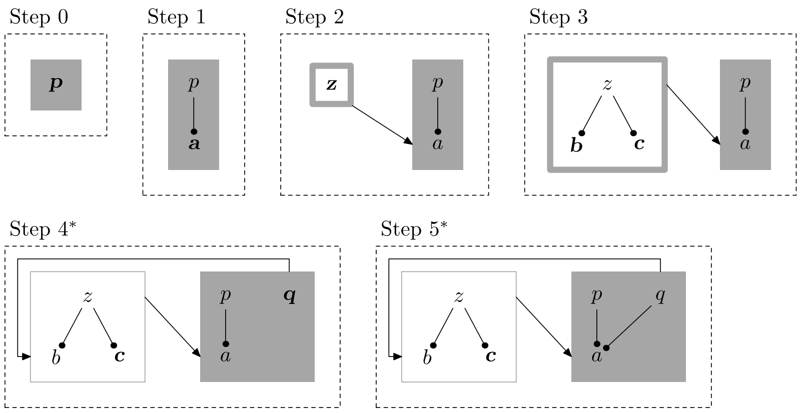


Fig. 20. GRN-graph-DS for p, which is not a GRN-graph-DD for Example 15.

Table 6
Alternative GRN-graph-DS for p, which is not a GRN-graph-DD, for Example 15.

Step	Case	$\mathcal{P}_i/\boldsymbol{O}_i$ (only $u(\boldsymbol{O}_i)$ shown)	$\mathcal{G}_i$	D_i/C_i
0	n/a	$(\{\boldsymbol{p}\}, \emptyset)$ $\emptyset$	$(\emptyset, \emptyset)$	$\emptyset$ $\emptyset$
1	1(ii)	$(\{p, \boldsymbol{a}\}, \{(p, a)\})$ $\emptyset$	$(\{p, a\}, \{(a, p)\})$	$\{a\}$ $\emptyset$
2	1(i)	$(\{p, a\}, \{(p, a)\})$ $\{(\{\boldsymbol{z}\}, \emptyset)\}$	$(\{p, a\}, \{(a, p), (z, a)\})$	$\{a\}$ $\emptyset$
3	2(ii)	$(\{p, a\}, \{(p, a)\})$ $\{(\{z, \boldsymbol{b}, \boldsymbol{c}\}, \{(z, b), (z, c)\})\}$	$(\{p, z, a, b\}, \{(a, p), (z, a)\})$	$\{a\}$ $\emptyset$
4*	2(i)(c)	$(\{p, \boldsymbol{q}, a\}, \{(p, a)\})$ $\emptyset$	$(\{p, q, z, a, b\}, \{(a, p), (z, a), (q, z)\})$	$\{a\}$ $\{b\}$
5*	1(ii)	$(\{p, q, a\}, \{(p, a), (q, a)\})$ $\emptyset$	$(\{p, q, z, a, b\}, \{(a, p), (z, a), (q, z), (a, q)\})$	$\{a\}$ $\{b\}$

Table 7
Attempted GB-dispute derivation for p.

Step	$\mathcal{P}_i$	$\mathcal{O}_i$	D_i	C_i
0	$\{p\}$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{a\}$	$\emptyset$	$\{a\}$	$\emptyset$
2	$\emptyset$	$\{\{z\}\}$	$\{a\}$	$\emptyset$
3	$\emptyset$	$\{\{b, c\}\}$	$\{a\}$	$\emptyset$
4	$\{q\}$	$\emptyset$	$\{a\}$	$\{b\}$
5	$\{a\}$	$\emptyset$	$\{a\}$	$\{b\}$
6	$\emptyset$	$\{\{z\}\}$	$\{a\}$	$\{b\}$
7	$\emptyset$	$\{\{b, c\}\}$	$\{a\}$	$\{b\}$
8	$\{q\}$	$\emptyset$	$\{a\}$	$\{b\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$$\begin{aligned} \mathcal{A} &= \{\, a, b \,\} \\ \mathcal{R} &= \{\, p \leftarrow q, \\ &\qquad q \leftarrow a, \\ &\qquad r \leftarrow p \,\} \\ \bar{a} = b, &\quad \bar{b} = r \end{aligned}$$

Now consider the ADM-graph-DD for p in Table 8 (shown in Fig. 21 graphically):

It is interesting to compare this ADM-graph-DD to the corresponding AB-dispute derivation of [12,14] (cf. Definition A.2 of Toni [27]). That dispute derivation runs as in Table 9. The two derivations are analogous, up until the transition from steps 4 to 5. At this point, the ADM-graph-DD effectively recognizes that p has already been encountered, and therefore

Table 8
ADM-graph-DD for p, for Example 16.

Step	Case	$\mathcal{P}_i$	$\boldsymbol{O}_i$ (only $u(\boldsymbol{O}_i)$ shown)	D_i	C_i
0	n/a	$(\{\boldsymbol{p}\}, \emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$
1	1(ii)	$(\{p, \boldsymbol{q}\}, \{(p, q)\})$	$\emptyset$	$\emptyset$	$\emptyset$
2	1(ii)	$(\{p, q, \boldsymbol{a}\}, \{(p, q), (q, a)\})$	$\emptyset$	$\{a\}$	$\emptyset$
3	1(i)	$(\{p, q, a\}, \{(p, q), (q, a)\})$	$\{(\{\boldsymbol{b}\}, \emptyset)\}$	$\{a\}$	$\emptyset$
4	2(i)(c)	$(\{p, q, \boldsymbol{r}, a\}, \{(p, q), (q, a)\})$	$\emptyset$	$\{a\}$	$\{b\}$
5	1(ii)	$(\{p, q, r, a\}, \{(p, a), (q, a), (r, p)\})$	$\emptyset$	$\{a\}$	$\{b\}$

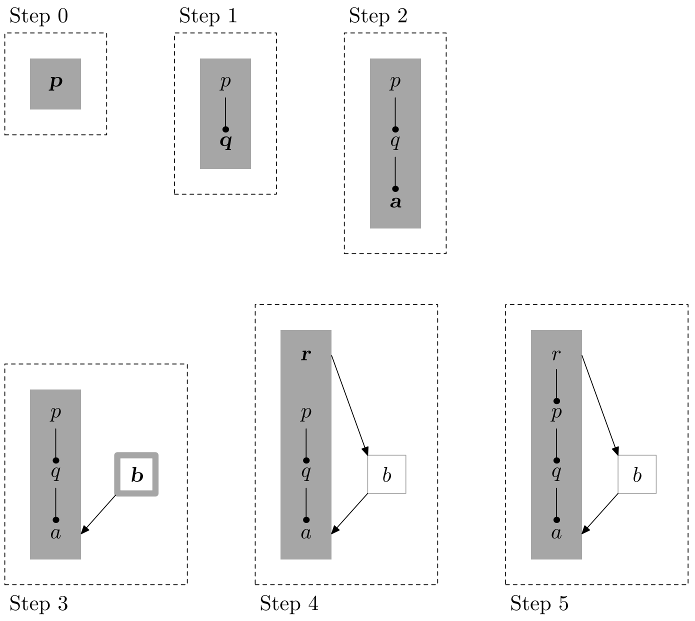


Fig. 21. ADM-graph-DD for Example 16.

Table 9
AB-dispute derivation for p, for Example 16.

Step	$\mathcal{P}_i$	$\mathcal{O}_i$	D_i	C_i
0	$\{p\}$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{q\}$	$\emptyset$	$\emptyset$	$\emptyset$
2	$\{a\}$	$\emptyset$	$\{a\}$	$\emptyset$
3	$\emptyset$	$\{b\}$	$\{a\}$	$\emptyset$
4	$\{r\}$	$\emptyset$	$\{a\}$	$\{b\}$
5	$\{p\}$	$\emptyset$	$\{a\}$	$\{b\}$
6	$\{q\}$	$\emptyset$	$\{a\}$	$\{b\}$
7	$\emptyset$	$\emptyset$	$\{a\}$	$\{b\}$

does not need to prove p again (by deriving q and then a). The *filtering* of the AB-dispute derivation, by contrast, only takes place on assumptions (achieved by the removal of those elements of rule bodies which are in D_i), and thus p and q are both proved again (at steps 5 and 6 of the AB-dispute derivation). In a short example such as that under consideration, this only makes a difference of two tuples saved, but in a less trivial example the gains in efficiency can be considerable. ⌟

Finally, we give an example of a GRN-graph-DD which shows the use of 'filtering by culprits'. In the case of the previous forms of derivation for tree-based arguments in ABA, this was not possible; our use of a cyclicity check, enabled by the use of the $\mathcal{G}_i$ component, allows us to introduce it.

Example 17. Consider the following ABA framework.

$$\mathcal{L} = \{\, p, q, r, s, a, b \,\}$$

$$\mathcal{A} = \{\, a, b \,\}$$

$$\mathcal{R} = \{\, p \leftarrow b, r,$$

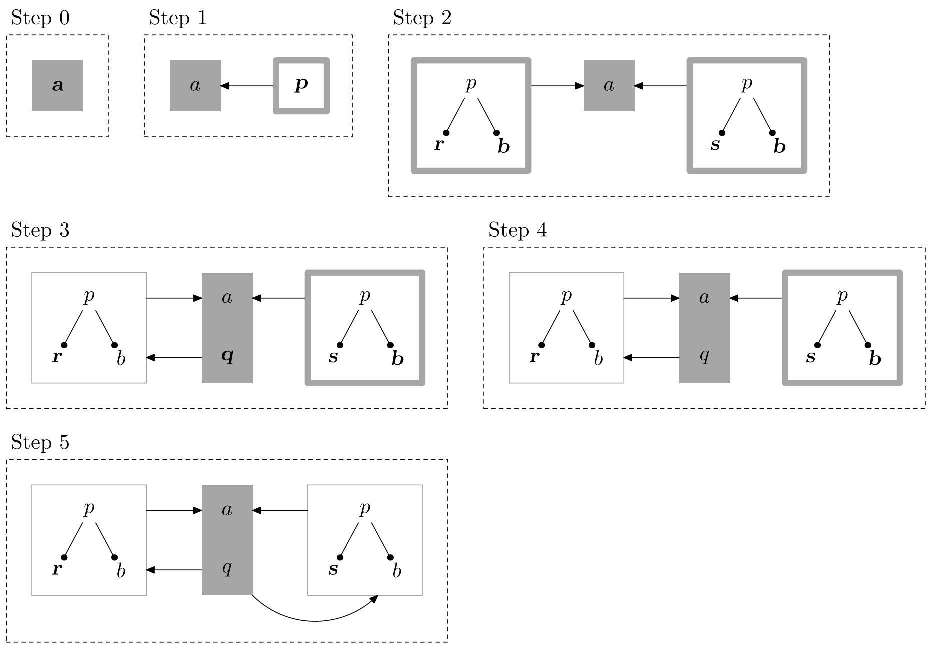


Fig. 22. GRN-graph-DD for Example 17.

Table 10
Tabulated form of derivation from Fig. 22.

Step	Case	$\mathcal{P}_i$	$\boldsymbol{O}_i$ (only $u(\boldsymbol{O}_i)$ shown)	$\mathcal{G}_i$	D_i	C_i
0	n/a	$(\{\boldsymbol{a}\}, \emptyset)$	$\emptyset$	$(\{a\}, \emptyset)$	$\{a\}$	$\emptyset$
1	1(i)	$(\{a\}, \emptyset)$	$\{(\{\boldsymbol{p}\}, \emptyset)\}$	$(\{a, p\}, \{(p, a)\})$	$\{a\}$	$\emptyset$
2	2(ii)	$(\{a\}, \emptyset)$	$\{(\{p, \boldsymbol{b}, \boldsymbol{r}\}, \{(p, b), (p, r)\}),$ $(\{p, \boldsymbol{b}, \boldsymbol{s}\}, \{(p, b), (p, s)\})\}$	$(\{a, p\}, \{(p, a)\})$	$\{a\}$	$\emptyset$
3	2(i)(c)	$(\{a, \boldsymbol{q}\}, \emptyset)$	$\{(\{p, \boldsymbol{b}, \boldsymbol{s}\}, \{(p, b), (p, s)\})\}$	$(\{a, p, q\}, \{(p, a), (q, p)\})$	$\{a\}$	$\{b\}$
4	1(ii)	$(\{a, q\}, \emptyset)$	$\{(\{p, \boldsymbol{b}, \boldsymbol{s}\}, \{(p, b), (p, s)\})\}$	$(\{a, p, q\}, \{(p, a), (q, p)\})$	$\{a\}$	$\{b\}$
5	2(i)(b)	$(\{a, q\}, \emptyset)$	$\emptyset$	$(\{a, p, q\}, \{(p, a), (q, p)\})$	$\{a\}$	$\{b\}$

$$p \leftarrow b, s,$$

$$q \leftarrow \}$$

$$\bar{a} = p, \bar{b} = q$$

A GRN-graph-DD for a is shown in Fig. 22. As usual, we also show the derivation in tabular form—see Table 10. At step 3, b was selected from the left-most opponent argument graph, and its contrary q added to $\mathcal{P}_3$; b is, accordingly, now a member of C_3, the set of culprits. In the transition from step 4 to step 5, according to case 2(i)(b), b is selected again; this time, since $b \in C_4$, there is no need to re-prove the contrary of b, q, since this has already been encountered. In the GB-derivations for tree-based arguments, this form of filtering using culprits is not possible: there may have been a loop in the dependencies which would make the derivation unsound. We avoid such loops using the $\mathcal{G}_i$ component, which is checked for cycles. ⌟

5.4. Soundness and completeness of graph-DDs

First, we give two soundness results for the derivations with respect to the argument graph semantics defined in §4.3.

Theorem 5.9. *For any X-graph-DD ($X \in \{$ADM, GRN$\}$) for s_0 with resulting argument graph $\mathcal{P}$, $\mathcal{P}$ is admissible and $s_0 \in v(\mathcal{P})$.* ⌟

Theorem 5.10. *For any GRN-graph-DD with resulting argument graph $\mathcal{P}$, there is some grounded argument graph G such that $\mathcal{P} \subseteq G$.* ⌟

Then, we give corresponding completeness results, again with respect to the argument graph semantics.

Theorem 5.11. *Let $\mathcal{L}$ be finite. If G is an admissible argument graph such that $s_0 \in v(G)$, then there is an* ADM*-graph-DD for s_0 with resulting argument graph some $\mathcal{P}$ such that $\mathcal{P} \subseteq G$.* ⌟

Theorem 5.12. *Let $\mathcal{L}$ be finite. If G is a grounded argument graph such that $s_0 \in v(G)$, then there is a* GRN*-graph-DD for s_0 with resulting argument graph some $\mathcal{P}$ such that $\mathcal{P} \subseteq G$.* ⌟

In the light of theorems shown in §4.3, we can show the following, concerning the soundness and completeness of graph-DDs with respect to admissible and grounded acceptance of sentences for tree-based arguments.

Corollary 5.13. *Let there be an* ADM*-graph-DD (respectively* GRN*-graph-DD) for $s_0 \in \mathcal{L}$ resulting in $\mathcal{P}$. Then the set of arguments represented in $\mathcal{P}$ is admissible (respectively an admissible subset of the grounded extension), non-bloated, and contains some* a *such that claim*(a) $= s_0$.

Proof. Directly from Theorems 4.20, 5.9, 5.10 and 4.12. □

Corollary 5.14. *Let $\mathcal{L}$ be finite. If* A *is an admissible (respectively grounded) extension with* a ∈ A *such that $s_0 =$ claim*(a)*, then there is an* ADM*-graph-DD (respectively* GRN*-graph-DD) for s_0 resulting in $\mathcal{P}$ such that support($\mathcal{P}$) $\subseteq$ support*(A).

Proof. Directly from Theorems 4.20, 5.11 and 5.12. □

5.5. Implementation

We implemented the X-graph-DDs in Prolog as `abagraph`.[7] Two principle factors prompted the choice of programming language. First, we wanted to conduct an experimental comparison of our algorithm with the leading implementation of dispute derivations for tree-based ABA, `proxdd`[8]—and `proxdd` is itself implemented in Prolog. Using the same language was therefore necessary for the fairness of the experiments. Secondly, Prolog itself has built-in backtracking; this made possible a relatively high-level encoding of the algorithm for X-graph-DDs. Of course, other languages could have been chosen. In previous work, for example, we have implemented the dispute derivations of [27] in parallel C++ [8], and such an approach would certainly have been feasible for argument graphs.

Given a representation of an ABA framework and some specific sentence $s \in \mathcal{L}$ as input, `abagraph` searches, in Prolog, to find all possible X-graph-DDs, $X \in \{\text{ADM}, \text{GRN}\}$, for $s \in \mathcal{L}$. As explained in §5.2, the definition of an X-graph-DS (and hence X-graph-DD) can allow for a given sequence to be continued in multiple different ways, depending on a strategy for the selection of various sets and members in the current tuple $(\mathcal{P}_i, \boldsymbol{O}_i, \mathcal{G}_i, D_i, C_i)$. In `abagraph` we provide the following built-in strategies for the various selections.

1. A priority ordering for the player choice. In general, the move from step i to step $i+1$ in Definition 5.5 might be made by the PROPONENT or OPPONENT. If the parameter, for instance, is OPPONENT < PROPONENT, this means that: if $\mathcal{P}_i$ is non-empty, then it will be selected, else if $\mathcal{P}_i$ is empty and $\boldsymbol{O}_i$ is non-empty, then $\boldsymbol{O}_i$ will be selected.
2. A selection criterion for the member $\mathcal{O} \in u(\boldsymbol{O}_i)$ (if step i is an OPPONENT step). Possible values here are:
   n the *newest* OPPONENT argument graph to have been added to $\boldsymbol{O}_i$ is selected;
   o the *oldest* OPPONENT argument graph to have been added to $\boldsymbol{O}_i$ is selected;
   s the OPPONENT argument graph G such that $|u(G)|$ is as low as possible is selected;
   l the OPPONENT argument graph G such that $|u(G)|$ is as high as possible is selected.
3. A selection criterion for the sentence from $u(\mathcal{P}_i)$, with possible values:
   n the *newest* sentence added to $u(\mathcal{P}_i)$ is selected;
   o the *oldest* sentence added to $u(\mathcal{P}_i)$ is selected;
   e select an assumption where possible (an *eager* strategy);
   p select a non-assumption (a sentence in $\mathcal{L} \setminus \mathcal{A}$) where possible (a *patient* strategy).
4. A selection criterion for the OPPONENT sentence from the selected $G \in u(\mathcal{O}_i)$, with values the same as those for (3).

There are 2 possible values for the first parameter, and 4 each for the remaining parameters, giving 128 possible strategies overall. Evidently, these are a very small portion of those strategies even quickly conceivable.

[7] Available from http://robertcraven.org/proarg/abagraph.html.
[8] See http://robertcraven.org/proarg/proxdd.html.

6. Experiments

6.1. Experiment design

We compared our graph-DDs, as implemented in `abagraph`, with the most competitive existing system (`proxdd`) for dispute derivations.[9] The implementation of the original algorithm (`proxdd`) uses the variant presented by Toni [27], which records the arguments as well as the attack relationships between them as they are constructed. This is appropriate for purposes of comparison, as our algorithm and its implementation (`abagraph`) record the full argument graph structures, including attacks and counter-attacks, as the derivations proceed. The work in the present section provides preliminary evidence that the dispute derivations we defined for argument graphs in §5 may offer computational advantages in speed and the number of queries answered over implementations of the standard dispute derivations for tree-based arguments. This can be seen as a first step towards a more thorough experimental evaluation, complementing the conceptual underpinnings (of argument graphs, their relation to tree-based arguments, the definition of a semantics for argument graphs, and the definition of sound and complete dispute derivations) that are the main focus of this paper.

For our experiments, we randomly generated ABA frameworks,[10] and compared the performance of each implementation on sample queries. The random generator takes as input a tuple of parameters, $(N_s, N_a, N_{rh}, N_{rph}, N_{spb}, N_{apb})$, as follows.

1. N_s is the total number of sentences in the framework, i.e., $|\mathcal{L}|$.
2. N_a is the number of assumptions. This can be given either as: (i) an integer; (ii) a percentage of the number of sentences; (iii) an interval $[min, max]$, where min and max are both integers; (iv) an interval $[p_{min}, p_{max}]$, where p_{min} and p_{max} both represent percentages of the number of sentences, $N_s = |\mathcal{L}|$. In cases (iii) and (iv), the implementation chooses a random number in the interval.
3. N_{rh} is the number of distinct sentences to be used as heads of rules. This parameter takes the same form of values as for parameter 2, above.
4. N_{rph} is the number of rules per distinct rule head, given as: (i) an integer; or (ii) an interval $[min, max]$, where min and max are integers. In case (ii), for each distinct rule head s, a random value n is chosen with $min \leqslant n \leqslant max$, and n different rules with head s are then added to $\mathcal{R}$.
5. N_{spb}, the number of sentences per body, given as: (i) an integer; (ii) a percentage of the number of sentences; (iii) an interval $[min, max]$, where min and max are both integers; (iv) an interval $[p_{min}, p_{max}]$, where p_{min} and p_{max} both represent percentages of the number of sentences, $N_s = |\mathcal{L}|$. In cases (iii) and (iv), the implementation chooses, rule-by-rule, a random number in the interval.
6. N_{apb}, the number of assumptions per body, given as: (i) an integer; (ii) a percentage of the number of sentences of the current body; (iii) an interval $[min, max]$, where min and max are both integers; (iv) an interval $[p_{min}, p_{max}]$, where p_{min} and p_{max} both represent percentages of the number of sentences in the current rule. In cases (iii) and (iv), the implementation chooses, rule-by-rule, a random number in the interval.

We presume the existence of the following subsidiary functions.

- $pickValue(X, Y)$, such that:
  - where $X \in \mathbb{N}$, $pickValue(X, Y)$ is X;
  - where X is a percentage value $P\%$, $pickValue(X, Y)$ is $X \times Y/100$;
  - where X is $[min, max]$ and $min, max \in \mathbb{N}$, then $pickValue(X, Y)$ is a random number in the interval $[min, max]$;
  - where X is $[min\%, max\%]$, then $pickValue(X, Y)$ is a random number in the interval $[Y \times min/100, Y \times max/100]$;
- if $N \in \mathbb{N}$ and S is a set, then $pickFrom(N, S)$ is a random subset of members of S such that $|pickFrom(N, S)| = N$;
- $randomMember(S)$, where S is a set, chooses a random member of S.

To produce a random ABA framework, we used Algorithm 6.1.

Systematic and thorough comparison of how the performance of our two implementations compares with different combinations of variation in these parameters would have taken a prohibitively long time; so we chose to perform experiments on four basic series of ABA frameworks. In the first three series, a single parameter varies on its own; in the final series, all parameters vary dependently. The form of parameters for all four series are shown in Table 11. In the first series of experiments, we varied the size of $\mathcal{L}$, while keeping all other parameters the same; since $|\mathcal{A}|$ was kept fixed at 15, this had the effect of varying the number of non-assumptions in the ABA framework. In the second series, we varied the number of rules per head, with the effect of varying $\mathcal{R}$. In the third series of experiments, we varied the number of sentences per rule; since the number of assumptions per rule body was not varied (it was given a random value in the interval $[0, 6]$) this mostly has the effect of varying the number of non-assumptions per rule. Finally, in the fourth series of experiments,

[9] Both available from http://robertcraven.org/proarg/.

[10] Code for the random generator, the ABA frameworks we used for the experiments, and experimental results are all available from http://robertcraven.org/proarg/experiments.html/.

Algorithm 6.1 *randomABA*($N_s, N_a, N_{rh}, N_{rph}, N_{spb}, N_{apb}$).

```
1: $\mathcal{L} := \{s_i \mid 0 \leqslant i < N_s\}$
2: $\mathcal{A} := \{s_i \mid 0 \leqslant i < pickValue(N_a, N_s)\}$
3: $\mathcal{R} := \emptyset$
4: for all $a \in \mathcal{A}$ do
5:     $\bar{a} = randomMember(\mathcal{L} \setminus \{a\})$
6: end for
7: $RH := pickFrom(pickValue(N_{rh}, N_s), \mathcal{L} \setminus \mathcal{A})$
8: for all $s \in RH$ do
9:     $RPH := pickValue(N_{rph}, 0)$
10:    while $RPH > 0$ do
11:        $SPB := pickValue(N_{spb}, N_s)$
12:        $APB := pickValue(N_{apb}, SPB)$
13:        $B := pickFrom(SPB - APB, (\mathcal{L} \setminus \mathcal{A}) \setminus \{s\})$
14:        $B := B \cup pickFrom(APB, \mathcal{A})$
15:        $\mathcal{R} := \mathcal{R} \cup \{s \leftarrow B\}$
16:        $RPH := RPH - 1$
17:    end while
18: end for
19: return $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\phantom{x}})$
```

Table 11
Experiment series design; all values are rounded to the nearest integer.

Series	Parameters	Values
1	$(N_s, 15, 20, [2,5], [0,6], [0,6])$	$N_s \in \{20, 30, 40, 50, 60, 70, 80, 90\}$
2	$(40, 15, 20, N_{rph}, [0,6], [0,6])$	$N_{rph} \in \{[2,5], [5,8], [8,11], [11,14], [14,17], [17,20], [20,23], [23,26]\}$
3	$(40, 15, 20, [2,5], N_{spb}, [0,6])$	$N_{spb} \in \{[0,3], [3,6], [6,9], [9,12], [12,15], [15,18], [18,21], [21,24]\}$
4	$(N_s, 37\%, N_s/2, [2, N_s/8], [0, N_s/7], [0, N_s/7])$	$N_s \in \{16, 24, 32, 40, 48, 56, 64, 72, 80, 88\}$

the size of $\mathcal{L}$ was varied, and the value of the other parameters tied to this. With the values of N_s given, this yields the sequence of parameters:

$$\begin{array}{ll}
(16, 6, 8, [2,2], [0,2], [0,2]), & (56, 21, 28, [2,7], [0,8], [0,8]), \\
(24, 9, 12, [2,3], [0,3], [0,3]), & (64, 24, 32, [2,8], [0,9], [0,9]), \\
(32, 12, 16, [2,4], [0,5], [0,5]), & (72, 27, 36, [2,9], [0,10], [0,10]), \\
(40, 15, 20, [2,5], [0,6], [0,6]), & (80, 30, 40, [2,10], [0,11], [0,11]), \\
(48, 18, 24, [2,6], [0,7], [0,7]), & (88, 33, 44, [2,11], [0,13], [0,13])
\end{array}$$

Underlying all four series of experiments are the parameters $(40, 15, 20, [2,5], [0,6], [0,6])$. Informal experimentation indicated that these values produce ABA frameworks with sentences which, when queried according to each of the semantics we study (admissible and grounded), yield answers with a mix between (i) immediate answers, yes or no; (ii) answers which failed to compute because resources were exceeded by the Prolog implementation, or because a time-limit was exceeded; and (iii) answers which were computed in times > 1 second, and which had nested structures of attack between the acceptable argument graph and those attacking it. We take the view that parameters producing frameworks with this mixture of sentences are desirable, since the effect of varying parameters on the proportion of queries falling into the different classes can then be studied.

An ABA framework only admits the possibility of circular arguments if it is not *p-acyclic*; this notion is defined in [14], and the definition is equivalent to:

Definition 6.1. Let the *dependency graph of* $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$ be the directed graph whose nodes are $\mathcal{L} \setminus \mathcal{A}$ and where there is a directed edge (s, s') iff there is a rule $s \leftarrow B \in \mathcal{R}$ such that $s' \in B$. $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$ is *p-acyclic* iff the dependency graph of $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$ is acyclic. ⌟

Thus, an indication of how our implementations fare in the presence or absence of circular arguments can be found by running experiments on p-cyclic (i.e., non-p-acyclic) and p-acyclic arguments. For each series of experiments, and each instantiation of parameters per series, 10 p-acyclic frameworks and 10 p-cyclic frameworks were randomly generated. This made for a total of 680 ABA frameworks in total. The division into p-cyclic and p-acyclic classes was motivated by a desire to investigate whether there were differences in performance between `abagraph` and `proxdd` in the two cases. Indeed, an important innovation in the graph-based approach is its completeness for grounded extensions, and associated means of preventing loops in the case of cyclical dependencies amongst sentences in ABA frameworks.

Table 12
Combined results for all types of derivation and cycle-type.

	`abagraph`	`proxdd`
Total exceptions	25	2572
Total exceptions and timeouts	4845	6830
Average non-zero solutions	12.947	213.798
Average time per query (secs.), exceptions excluded	41.632	44.934

Table 13
Results according to derivation type.

	$X = \text{ADM}$		$X = \text{GRN}$	
	`abagraph`	`proxdd`	`abagraph`	`proxdd`
Total exceptions	0	928	25	1644
Total exceptions and timeouts	2359	3215	2486	3615
Average non-zero solutions	21.281	356.098	3.714	5.696
Average time per query (secs.), exceptions excluded	40.964	45.501	42.305	44.289

In comparing the results of the two implementations, that of the structured X-derivations of Definition 6.3 of [27] and the graph-DDs of Definition 5.7, it is important to set the same search strategy in each case. Each possible search strategy for `abagraph` is represented by a choice of the four parameters described in §5.5. The parameters for `proxdd` are largely similar. Our method was as follows. We first selected 10 random strategies to be used throughout the experiments. Then, for each framework, we paired a random sentence s with each strategy, and queried both `proxdd` and `abagraph` to find admissible and (fragments of) grounded extensions containing an argument whose claim is s. In each case, all solutions were attempted to be found. We imposed a time-out of 120 seconds on each computation (each query/implementation/semantics triple). For each computation, we recorded: (i) the time taken; (ii) whether or not the computation timed out, threw an exception because resources were exceeded, or successfully completed; (iii) how many solutions were found. If the computation terminated successfully, then the solutions found are all solutions possible; if there was no successful termination—through time out or an exception—then the number of solutions found may not be the total possible. In sum, this represented a maximum of 1360 hours of computation (a little over 56 days). Experiments were conducted on a series of HP Compaq dc8200 machines, with an Intel Core i7-2600 3.40 GHz processor and 8 GB of RAM, running 64-bit Ubuntu 13.04; the Prolog implementation was SICStus 4.2.3.

6.2. Results

We first give total statistics across the different series of experiments, in tabular form, followed by results showing the queries successfully answered by an individual solver uniquely. We then give series-specific results (which concern quantities for the different steps of each series) as graphs.

In the three tables which follow, we record the following information:

- *Total exceptions*, where the exceptions thrown were always due to exceeding the memory resources of the Prolog system used.
- *Total exceptions and timeouts* combined, where the timeout was, as mentioned above, 120 seconds.
- *Average non-zero solutions*. This is calculated by averaging the number of solutions for those queries which completed successfully (i.e., disregarding the solutions found in the case where the result was a timeout), *if* that number was non-zero. This number, when compared across `abagraph` and `proxdd`, therefore gives an indication of the number of 'redundant' or otherwise conceptually 'bad' solutions—in the senses discussed in §3—which are eliminated by `abagraph` through the shift to argument graphs.
- *Average time per query* was calculated for those queries in which neither implementation threw an exception. (Note that if exceptions were included, then the results here tend to favour `abagraph` even more.)

Table 12 shows the combined results for all queries, no matter what the type of derivation, or whether the frameworks were p-cyclic or p-acyclic. There is a clear advantage to using `abagraph` because the number of exceptions is dramatically lower. The time taken to return answers to queries is also lower, though here the gains are smaller. Finally, the fact that the use of argument graphs results in much fewer solutions overall, shows that the number of 'redundant' solutions (in the sense mentioned previously) is, using standard tree-based arguments, relatively high.

Next, we split the results according to whether the derivation was for the admissible or grounded semantics. The results are shown in Table 13. A number of aspects of this are worthy of comment. First, in the case of ADM-graph-DDs there were no exceptions. We conjecture that the added overhead of a non-empty $\mathcal{G}$ component in the case of GRN-graph-DDs can be used to explain the higher number of exceptions. Secondly, the difference between the number of solutions found by `proxdd` and `abagraph` is much higher in the case of ADM-graph-DDs than with GRN-graph-DDs. This is to be expected:

Table 14
Results according to presence of cycles.

	p-cyclic		p-acyclic	
	abagraph	proxdd	abagraph	proxdd
Total exceptions	21	1858	4	714
Total exceptions and timeouts	2896	4323	1949	2507
Average non-zero solutions	18.716	2.738	9.109	281.500
Average time per query (secs.), exceptions excluded	48.281	55.614	34.555	35.481

there is a single grounded extension in the case of standard, tree-based semantics for $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{})$, of which the set of arguments found by `proxdd` in a grounded dispute derivation must be an admissible subset; yet there can often be many grounded argument graphs, giving rise to a larger number of solutions for the latter. Other aspects of the comparison are broadly consonant with the merged results for ADM-graph-DDs and GRN-graph-DDs presented previously.

Finally, we present in Table 14 results which consider different types of input frameworks, according to whether these were p-acyclic or not (as verified by a simple graph-theoretic analysis in Prolog on each framework). What is most worthy of comment here is the comparison on the number of non-zero solutions found for p-cyclic frameworks: the number found by `abagraph` is much higher than that found by `proxdd`. Why should this be? First note that, in fact, this tendency is much more marked for ADM-graph-DDs.[11] This is as we expected: grounded extensions are unique, and this low cardinality tends to carry through to the number of argument graphs, and then to the associated number of solutions for queries. In the p-cyclic case, ADM-graph-DDs give an average of 32.569 solutions for `abagraph`, and 3.682 solutions for `proxdd`; GRN-graph-DDs give an average of 2.432 solutions for `abagraph`, and 1.408 for `proxdd`. The explanation is, in fact, simple: there are relatively few instances where `abagraph` finds a high number of solutions for ADM-graph-DDs queries: 29 instances where the number of solutions is higher than 20. Yet of those 29 queries, all save one, when posed to `proxdd`, resulted in either timeout or an exception; and thus they are not counted in the finding of the average number of solutions.

Another measure which can help in comparing the performance of `abagraph` against `proxdd` on our particular sample data is that of the 'unique solver contribution' made by each system in answering queries.[12] For this, we used the results for all experiments, dividing them into those for p-cyclic and p-acyclic frameworks, and dividing each of those two categories further into results obtained for `abagraph`, and results obtained for `proxdd`. This gave four groups. We excluded queries which had failed to be answered within the time-out of 120 seconds, and those queries which raised exceptions in either system, thus keeping only the results for completely answered queries for both systems. Then, for steps of intervals of 6 seconds, and for each of the four groups, we recorded the number of queries answered uniquely by the system in question (`abagraph` or `proxdd`) in the amount of seconds or less—first, the number of queries answered in less than 6 seconds, then those answered in under 12 seconds, etc., up to the maximum of 120 seconds. The results are shown in Fig. 23. They provide preliminary evidence that `abagraph` may find a larger subset of query answers, with a more marked difference in the case of frameworks with cycles. For example, after 30 seconds, `abagraph` had found 1425 solutions for queries to p-cyclic frameworks which `proxdd` had not found, and 570 solutions for queries to p-acyclic frameworks which `proxdd` had not found. Still after 30 seconds, `proxdd` had found only 1 solution to a query for a p-cyclic framework that `abagraph` had not found, and 19 solutions to queries for p-acyclic frameworks that `abagraph` had missed.

In presenting the results for sequences of parameters, we will show the time taken on average for answering a query, where this is understood to be the time taken to find all possible answers. Thus, for some $s \in \mathcal{L}$, there may be multiple ADM-graph-DDs or GRN-graph-DDs for s, giving rise to many different $\mathcal{P}_n$ which are admissible argument graphs or (subgraphs of) grounded argument graphs. Further, as explained above, there are several possible outcomes for each query: either (i) all answers were found in under 120 seconds (the chosen timeout); (ii) the query fails to complete before the timeout; or (iii) an exception is raised. Those queries for exceptions were raised for either system, i.e., those of class (iii), were removed from contributing to the results. (However, it is notable that `proxdd` produced by far the greater number of exceptions for the particular queries and frameworks we used.) In comparing the results of the two systems, we show the results for ADM-graph-DDs and GRN-graph-DDs separately, side by side; intuitive inspection of the shape of the graphs in each case reveals that the results for each series are similar regardless of whether the derivation is ADM or GRN. The different 'steps' shown on the x-axis of the graphs represent the various increasing parameters for the random frameworks generated, according to Table 11.

First, Fig. 24 shows the results for series 1, in which $|N_s|$ (i.e., $|\mathcal{L}|$) is gradually increased as all other parameters remain the same. Aside from the peak at step 2, the tendency here is for the average time for a query gradually to decrease as $|\mathcal{L}|$ increases. It might be thought that the explanation for this is the fact that, as $|\mathcal{L}|$ gets larger without any other parameters being affected, the likelihood increases that sentences in $v(\mathcal{P}_i)$, at any stage of an X-graph-DS, will not be the head of a rule—and so that as $|\mathcal{L}|$ increases the number of admissible argument graphs containing s must diminish. It is apparent that

[11] The different figures for ADM-graph-DDs and GRN-graph-DDs, for the p-cyclic and p-acyclic cases, are not shown in Table 14, but were recorded in experiments.

[12] We thank an anonymous reviewer for prompting us to consider this measure and include results for it in the paper.

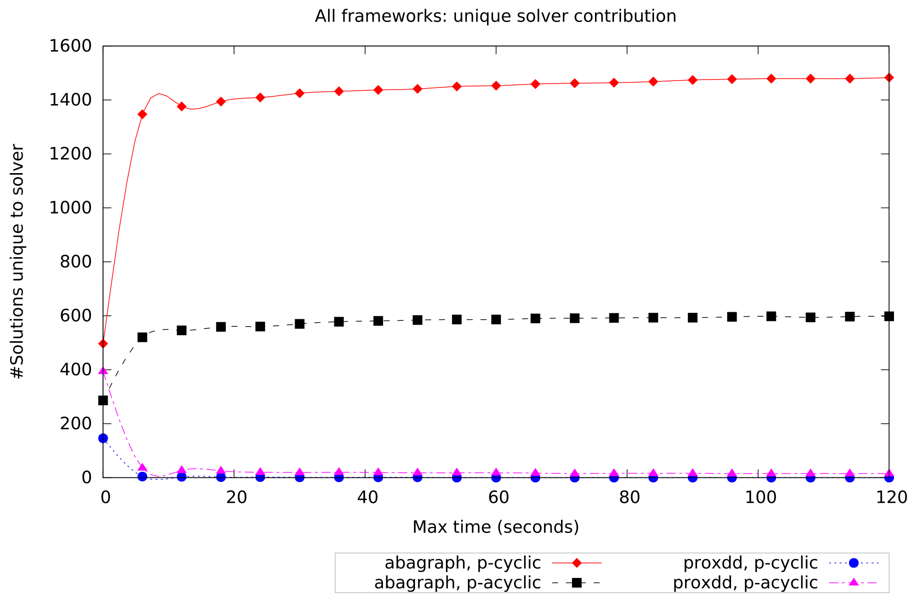


Fig. 23. Unique solver contribution, per framework type, p-cyclic and p-acyclic. For each time step on the x-axis, the number of queries uniquely solved by the given solver is shown on the y-axis.

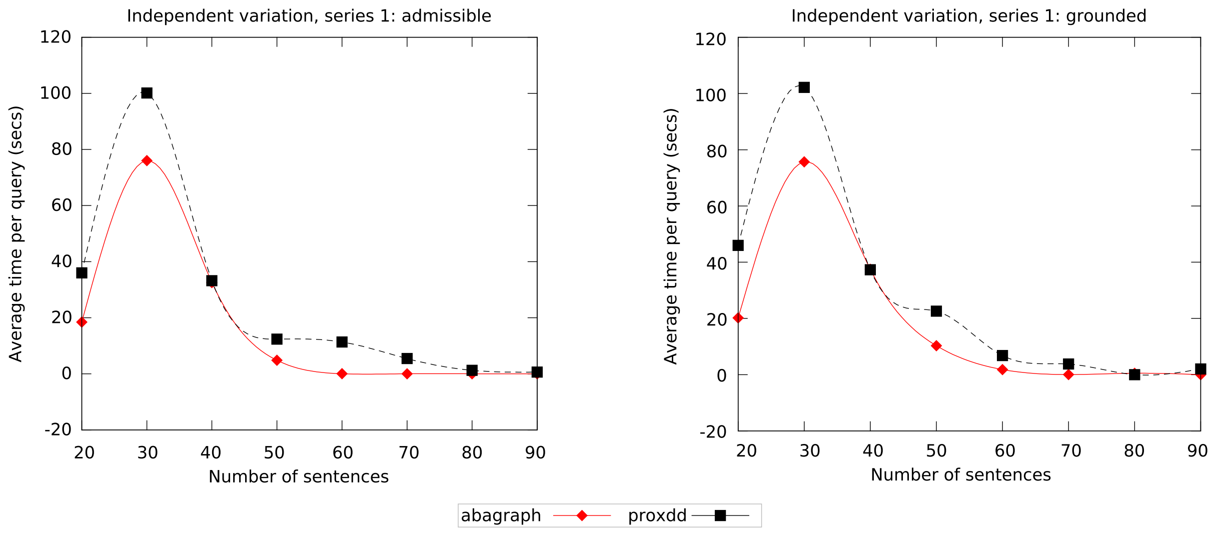


Fig. 24. Query times for randomly-generated frameworks in which the language-size, or number of sentences $|\mathcal{L}|$, is gradually increased, while all other parameters for the framework generation are held fixed.

this is not the reason, since the number of solutions does not gradually diminish as $|\mathcal{L}|$ increases. We leave more detailed analysis for future work.

Fig. 25 shows series 2, in which the number of rules per given rule head is gradually increased. This conforms to expectation, in that as the number of rules per head increases, it is likely that the number of possible argument graphs and arguments to be explored, when a given $s \in \mathcal{L}$ is chosen, will naturally increase. Though `abagraph` outperformed `proxdd` in the particular experiments we ran, this is not by very much; and the tendency in these experiments was for the performances of the two systems to become increasingly closer as the number of rules per head gets larger. We conjecture that the reason for this is the presence of the acyclicity check on the $\mathcal{P}_i$ and $\mathcal{G}_i$ components, which in general becomes more demanding as the number of rules per head increases; we leave possible confirmation of this for further work.

Thirdly, series 3 gradually increases the number of sentences per body, with results shown in Fig. 26. This shows very little effect on the performances of `abagraph` and `proxdd`, with `abagraph` consistently outperforming `proxdd` on queries for the particular frameworks we generated. We ourselves find it counter-intuitive that the number of sentences per body, when increased, should have little impact on the time taken to find solutions: our expectation had been that more

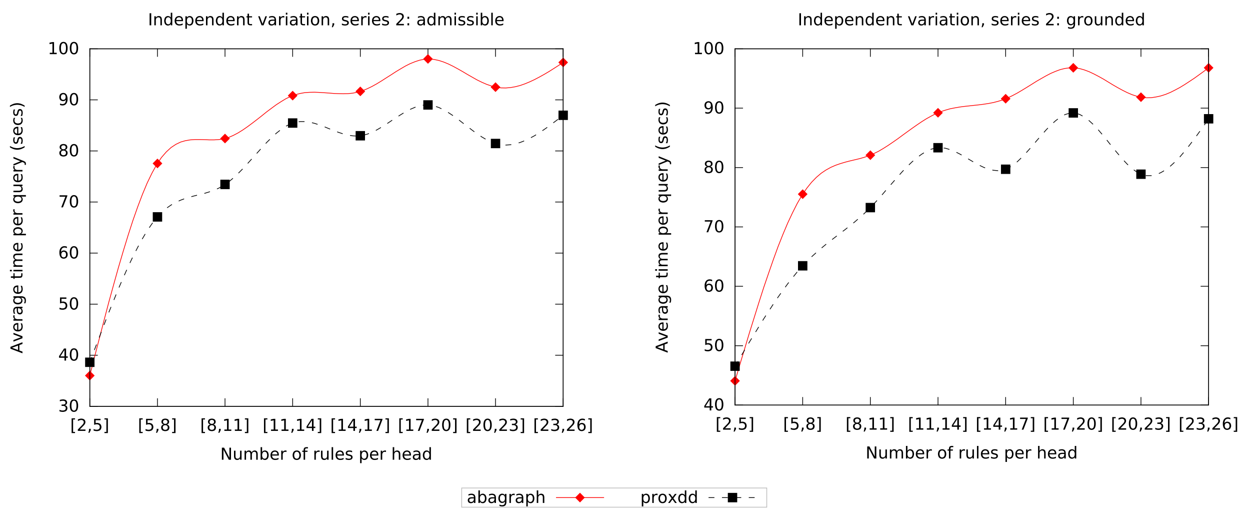


Fig. 25. Query times for randomly-generated frameworks in which the number of rules per head is gradually increased. The *x*-axis shows the interval within which the number of rules per head for each non-assumption that *is* the head of rules must fall (N_h, see §6.1). Other parameters are held fixed.

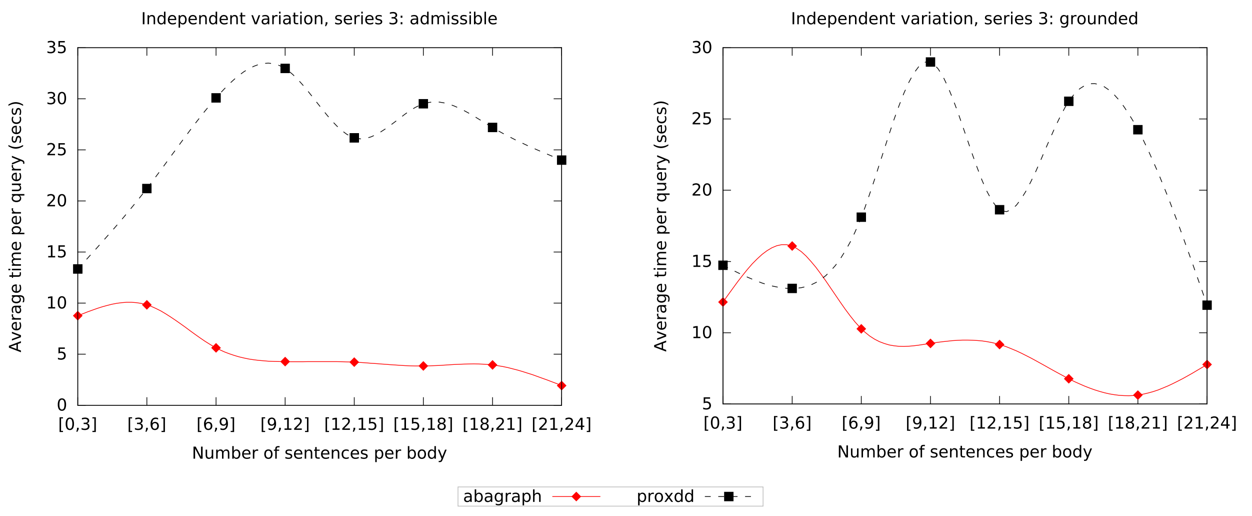


Fig. 26. Average query times as the number of sentences per rule body is gradually increased; for each body, the number of sentences is randomly selected within the given interval.

sentences per body would required more to be proved in order to establish a given argument graph or argument. We again leave deeper analysis for future work.

Finally, we consider the average time for answering queries where the parameters for creating random frameworks vary together (series 4). These are shown in Fig. 27. Though `abagraph` outperformed `proxdd` here, the tendency is for this to be by an increasingly small margin, as the size of the framework increases. As for series 2, we conjecture that the reason for the approach of the two curves to each other is the acyclicity checks on $\mathcal{P}_i$ and $\mathcal{G}_i$; we leave further investigation for future work.

From this experimentation we draw the tentative conclusion that, in general, there may be computational gains to using `abagraph` and argument graphs over `proxdd` and tree-based arguments. These gains are mostly seen in the number of solutions produced and the likelihood of a full set of answers being found at all; but there my also be also minor gains in the time for a computation.

As Tables 12–14 show, the number of non-zero solutions found by `abagraph` for the experimental data we used was much lower than that found by `proxdd`; we noted that these figures are a measure of the 'redundant' or 'bad' solutions excluded by `abagraph`. In future work, we would like to compare the experimental results achieved for `abagraph` with

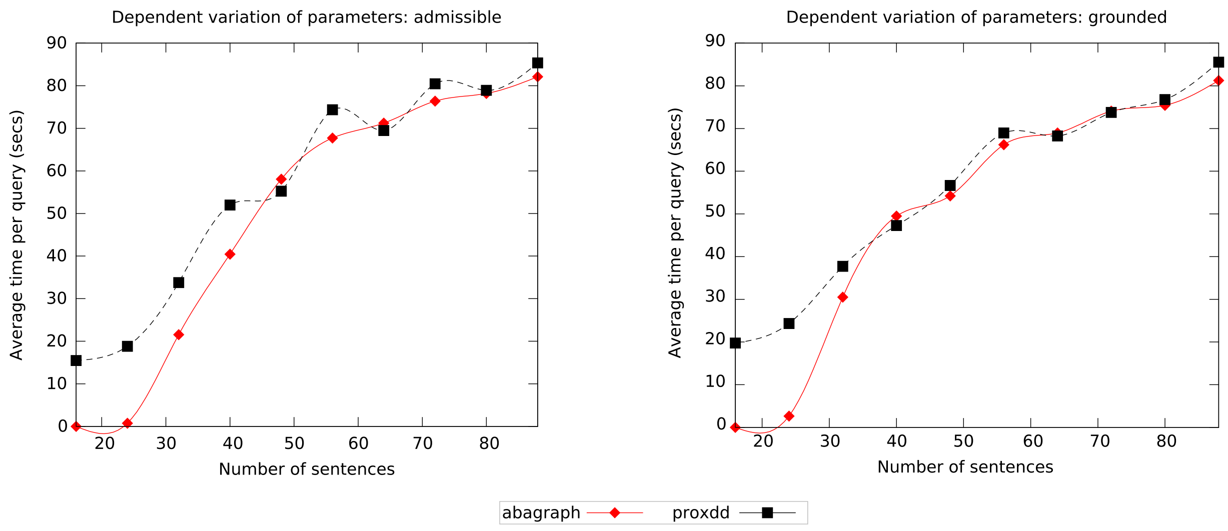


Fig. 27. Average query times, as all parameters are varied dependently. On the x-axis is shown, representatively, $|\mathcal{L}|$. The way the other parameters for framework generation are varied is described in §6.1.

a theoretical analysis of the optimum number of distinct solutions found for a given selection of frameworks, thus seeing whether the number of non-zero solutions found by `abagraph` is, or is near to, an optimum.[13]

7. Related work

As mentioned in the introduction, the current paper substantially extends the work we first presented, with a co-author, in [9]. In that paper, we introduced the notion of *rule-minimality* as it applies to the tree-based arguments which are standard in ABA (it could also easily be extended to other forms of structured argumentation). In the current paper, by contrast, we move to a more economical and, arguably, conceptually preferable use of argument graphs as the fundamental representation of arguments. As §§4.1–4.2 of the current paper show, focused argument graphs are in one-one correspondence with rule-minimal arguments—but an argument graph may also represent a set of rule-minimal arguments, where that set is non-bloated. Since non-bloatedness can be motivated in much the same way as rule-minimality, then this capacity is an advantage. Our previous work in [9] was also restricted to the case of the grounded semantics; here we also cover the admissible semantics, and include full definitions and proofs for every notion and theorem, at the same time adding completeness results for both the admissible and grounded semantics.

Our work on the completeness of the grounded semantics has been influenced by that of Thang et al. [26]. In that paper, the authors define a dialectical proof-procedure to compute admissible subsets of the grounded extension of an abstract argumentation framework. Significantly, the use of a dependency graph over arguments, grown gradually during the proof-procedure, ensures that the proof-procedure terminates finitely. In our work, we adapt the use of such a graph to the case of ABA—it forms the $\mathcal{G}_i$ component of the tuples given in Definition 5.5. Incorporating this into the definition of a GRN-graph-DD allows us to prove the completeness of our procedures in Theorem 5.12; the adaptation is a matter of paying attention to the internal structure of arguments, so that the $\mathcal{G}_i$ is defined over sentences instead of arguments. Whereas previous completeness results for dialectical proof-procedures for ABA [12,14,27] were restricted to p-acyclic frameworks [14], our use of a graphical dependency graph $\mathcal{G}_i$ and associated acyclicity check enables us to prove completeness more generally, without the previous restriction.

The overall structure of the X-graph-DSs given by Definition 5.5 follows a common blueprint which was established in [12] and [14]. As in Definition 6.3 of [27] (the definition of a 'structured dispute derivation'), our graphical derivations retain the structure both of the PROPONENT's justifications (through the argument graphs $\mathcal{P}_i$), and the various OPPONENT argument graphs which may attack the PROPONENT's claims (which are kept in $\boldsymbol{O}_i$). As we discuss in §5.3, the current paper introduces improved forms of filtering over the previous dispute derivations for ABA. Cases 1(ii) and 2(ii) of Definition 5.5, with their acyclicity conditions, ensure that, in the expansion of an argument graph, no sentence is proved twice. In the case of the PROPONENT, this is also ensured by case 2(i)(c), where a sentence is added as unmarked to $\mathcal{P}_{i+1}$ only if it has not already been added. In previous work, the PROPONENT could repeat lengthy justifications for sentences. Similarly, suppose that $a \in \mathcal{A}$ is part of the PROPONENT's support. In previous dispute derivations for ABA, if OPPONENT arguments for $\bar{a}$ have been computed already, those same arguments might later be recomputed if $\bar{a} = \bar{b}$ for some $b \in \mathcal{A}$ also in the PROPONENT support (for $a \neq b$). We improve on this in case 1(i) of Definition 5.5, where we add new potential argument graphs to $\boldsymbol{O}_{i+1}$

[13] We thank an anonymous reviewer for suggesting this further work to us.

only when they have not already been added. Some of these forms of filtering are extrapolations, to the case of ABA, of the use of the SP_i and SO_i components to achieve filtering in Definitions 4 and 8 of [26].

Modgil and Prakken [23] define a form of structured argumentation, ASPIC$^+$, with both strict and defeasible rules, and with notions both of a *contrary* and *contradictory* defined, as well as several forms of semantics. Arguments in ASPIC$^+$ are expressions defined using the rules and an underlying logical language; although no notion of tree is explicitly invoked in the definition of an argument, it is clear that ASPIC$^+$ arguments might easily have been defined as trees, and that notions of rule-minimality, bloatedness, circular and flabby arguments, would apply. The formalism allows arguments to be circular and flabby, and extensions to be bloated, and in this respect is similar to standard presentations of ABA. An interesting direction for future work is the study of argument graphs in the context of ASPIC$^+$.

García and Simari [20] take a logic-programming approach to defeasible argumentation, DeLP, also employing sets of strict and defeasible rules in the construction of arguments. DeLP, similarly to other forms of argumentation, also employs a notion of dialectical tree in order to establish whether a given claim is to be accepted (i.e., whether it is 'undefeated'), or not; nodes of the tree are labelled by *argument structures* (R, l), where l is a literal and R a set of defeasible rules which can be used, with the strict rules, to derive l. As opposed to our argument graphs, the structures (R, l) do not record the structure of a derivation. However, the sets R are required to be $\subseteq$-minimal, in that no smaller $R' \subset R$ could be used to derive the same literal l. As we note in the current paper, this constraint means that the construction of arguments is potentially computationally expensive, compared to our top-down, graph-based, rule-minimal approach. Another aspect of DeLP is that where an argument structure (R, l) has already appeared in a dialectical tree, no argument structure (R', l') such that $R' \subset R$ may appear, further on the same path as (R, l) from the root. We impose an analogous constraint on the PROPONENT argument graphs $\mathcal{P}_i$, in that once some $s \in \mathcal{L}$ is in $v(\mathcal{P}_i) \setminus u(\mathcal{P}_i)$, it can never be the case, for $j > i$, that $s \in u(\mathcal{P}_j)$. Since our use of argument graphs imposes that, for a given sentence s, only one rule with head s is ever used to prove s, it must be that, for any rule $s \leftarrow s_1, \ldots, s_n$, if $s, s_1, \ldots, s_n \in v(\mathcal{P}_i) \setminus u(\mathcal{P}_i)$, then that rule is never selected again for the (PROPONENT) case 1(ii) of Definition 5.5. This corresponds to the constraint on sub-argument structures in DeLP.

Vreeswijk [29] defines arguments recursively so that a given argument depends on its subarguments. An argument can be a sentence; or a structure $\sigma_1, \ldots, \sigma_n \rightarrow \phi$ (a strict rule) or $\sigma_1, \ldots, \sigma_n \Rightarrow \phi$ (a defeasible rule), such that ϕ is a sentence and $\sigma_1, \ldots, \sigma_n$ are subarguments. Of particular interest is the fact that Vreeswijk imposes a constraint so that any sentence in an argument cannot depend, through the strict or defeasible rules, on another occurrence of itself elsewhere in the argument (see his Definition 2.5, p. 232); to this degree he shares part of our motivation. However, Vreeswijk's definition of argument still allows them to be flabby, and sets of arguments can accordingly be bloated. In effect, trees are retained to structure arguments. In our work, by contrast, we shift to a different underlying representation, and remove the three forms of undesirability at once.

Our concern with the efficient representation and calculation of arguments is shared with others. Besnard and Hunter [4] require arguments to have a minimal support; the analogous notion for us (minimal sets of assumptions as support) is not implied by switching to argument graphs as the fundamental representation. Support-minimality needs to be ascertained 'globally', by checking the entire framework for alternative argument graphs. The property of rule-minimality which, as we showed is Section 4.1, characterizes those tree-based arguments which are straightforwardly equivalent to argument graphs, is close to condition 3 in the definition of argument structure (Def. 3.1) in [20]. Efstathiou and Hunter [16] present an approach for generating arguments within the context of logical argumentation, using *connection graphs* [21] to optimize the search for arguments, where—as with [4]—the support for a given claim consists of a minimal set of propositional formulas which together imply the claim.

There are many implementations of different forms of argumentation; we here compare our work with two other representative systems, one for abstract argumentation and one for structured argumentation. For abstract argumentation, ASPARTIX [17] uses an underlying answer-set solver and encodes the argumentation framework and rules defining various semantics, in order to find extensions. Our work in the current paper takes a 'top-down' approach, starting with a given sentence s and incrementally constructing arguments and attacks only when relevant; in order to answer the question of whether a given $s \in \mathcal{L}$ is contained in a grounded argument graph, we may not need to construct the entire grounded argument graph, but only an admissible subgraph thereof.

Snaith and Reed [25] present TOAST—an online, java-based implementation of the ASPIC$^+$ framework. Rules, preferences over rules, assumptions, and other elements of the ASPIC$^+$ framework can be entered, and a query is a specific formula. An underlying engine then evaluates, for several different semantics, whether there is an acceptable extension which supports the given formula. The implementation works by constructing the entire abstract argumentation framework from the entered rules and language; this again contrasts with our top-down approach which only considers relevant information. Plainly, the different approaches may, in this respect, be suited to different forms of application.

8. Conclusion

In this paper we have proposed an alternative representation for the rational dependencies among sentences in assumption-based argumentation (ABA): argument graphs. An argument graph is analogous both to a single traditional, tree-based argument in ABA, and also to a set of such arguments. In so doing, it constrains the rational structures represented to be free from certain properties possible in the tree-based representation—the properties of circularity, flabbiness, and bloatedness. We argued these correspond to nothing possible or desirable in the case of the relations of support identified by a rational agent.

We showed that the existence of our argument graphs corresponds both individually, and at the level of extensions in the admissible and grounded semantics, to the existence of arguments and extensions in the tree-based approach. We defined new notions of attack and, on that basis, new notions of semantics for argument graphs. We also introduced two kinds of graph-based dispute derivations, which we proved to be sound and complete with respect to the admissible and grounded semantics of such argument graphs respectively. Thus, although we believe that argument graphs are a superior representation of the structures which an agent brings to bear in argumentation, the argument-graph approach may also be used, computationally, as a leaner way of proving admissibility or groundedness of a sentence with respect to the traditional extension-based semantics. Another advantage of our work here is a completeness result for derivations.

We also conducted experiments to investigate the computational efficiency of our new approach, and concluded that although there is a consistent minor advantage of speed in the case where tree-based and graph-based approaches both terminate, there appear to be many more instances of tree-based derivations where the existing best implementation fails owing to memory resources having been exceeded. Thus there may be computational benefits in switching to argument graphs.

The graph-DDs we define borrow from the work of Thang et al. [26] the use of a graph whose acyclicity is an essential prerequisite of success. However, whereas Thang et al. [26] provide a computational machinery for abstract argumentation [11], we have focused on structured argumentation in the form of ABA. Moreover, Thang et al. [26] consider several argumentation semantics; we have focused on the admissible and grounded semantics.

We see many directions for future work. In the present paper we have taken ABA as the basic formalism, and applied the graph-based approach to it. However, as noted, tree-based arguments are widely used in many forms of structured argumentation. We believe that the motivation for the present work—that of outlawing circularity, flabbiness and bloatedness—applies just as much to these other forms of structured argumentation, and would support converting to argument graphs as the underlying representation. We intend to investigate this further, with respect to the structured argumentation frameworks of [23,20,4,1].

We believe that an absence of rule-minimality in tree-based approaches to argumentation should not confusedly be thought to correspond to the representation of aggregated reasons for belief. In the present paper we have not attempted to incorporate aggregation into the graph-based approach. However, this is evidently of crucial importance. We therefore intend to broaden our work to allow for aggregation.

In the present work we have introduced new graph-based semantics, defining notions of admissible, complete and grounded argument graphs. Our main focus was on establishing fundamental concepts, relating them to traditional approaches in ABA, and in showing the computational efficacy of our approach. However, it would be of great interest to conduct a more comprehensive investigation into forms of semantics possible with argument graphs. That would involve looking at equivalents for other leading semantics of argumentation—preferred, stable, semi-stable, ideal extensions [11,7,14]—as well as investigating whether the use of argument graphs allows other forms of semantics which have no direct equivalent in the use of sets of tree-based arguments.

The computational complexity of several reasoning tasks in ABA, e.g., that of determining membership in admissible and grounded extensions, or the existence of extensions, is known [10,15]. An interesting question for future work is whether the computational complexity of the same reasoning tasks changes when argument graphs rather than sets of argument trees are adopted.

We have focused on *flat* ABA frameworks. Although they are of restricted form, these admit several interesting instances, notably logic programming and default logic [6]. Another interesting topic for future research is the extension of our novel semantics and procedure to the case of *non-flat* ABA frameworks, as defined in [6].

Finally, we intend to investigate further the properties of our implementation of the graph-DDs we have defined, and to see whether the performance can be improved. As noted in §6, there are patterns in the comparison of `abagraph` with `proxdd` that deserve additional study; we also wish to implement any derivations for alternative semantics that we will define. There are also several respects in which we wish to make the experimental analysis of the performance of `abagraph` more thorough. First, it would be desirable to vary parameters for random framework generation which relate to the contrary of assumptions, such as how many different contraries there are, and the distribution of contraries amongst $\mathcal{R}$. We also want to conduct experiments on real-world data, in order to see whether the indications of respects in which `abagraph` gives a better performance are borne out on realistic problem areas. It would also be desirable to obtain measures, for some of the randomly-generated ABA frameworks we have used, on the 'ideal' number of solutions for given queries, in order to test how far `abagraph` approximates to that idea.

Appendix A. Proofs

Note that in the following proofs, we sometimes write an argument graph G in the form of the pair $(v(G), e(G))$. Moreover, for graphs G, G', we use $G \cup G'$ to denote the graph $(v(G) \cup v(G'), e(G) \cup e(G'))$. For convenience, we recall the statement of the theorem before each proof.

A.1. Theorem 4.6

(i) Let a be a non-circular, non-flabby argument. Then there is a unique graphical conversion G of a which is a focused argument graph with $claim(\mathsf{a}) = claim(G)$ and $support(\mathsf{a}) = support(G)$, such that a is represented in G. (ii) Let A be a non-bloated set of arguments. Then there is a unique graphical conversion G of A with $claims(\mathsf{A}) \subseteq v(G)$ and $\bigcup_{\mathsf{a}\in\mathsf{A}} support(\mathsf{a}) = support(G)$ and each argument in A is represented in G.

Proof. For (i), define G to be such that

$$v(G) = labels(nodes(\mathsf{a})) \setminus \{\top\}$$

$$e(G) = \{(s, s') \mid s' \neq \top, \exists n, n' \in nodes(\mathsf{a}), label(n) = s, label(n') = s', n' \in children(n)\}$$

We first show that G is an argument graph (cf. Definition 4.1). Evidently $v(G) \subseteq \mathcal{L}$. Now, first, if $s \in v(G) \cap \mathcal{A}$, then there can be no nodes n, n' in a with $n' \in children(n)$ such that $label(n) = s$; so $s \in sinks(G)$. Secondly, suppose $s \in v(G) \setminus \mathcal{A}$. Then there is at least one non-leaf node n in a labelled by s, such that there is a rule $s \leftarrow s_1, \ldots, s_m$ in $\mathcal{R}$, and the labels of the children of n are $s_1, \ldots, s_m$. These must be all the labels of the children of any node labelled by s, apart from $\top$; for otherwise $\{\mathsf{a}\}$ would be bloated, and then by Theorem 3.3, a would not be non-circular and non-flabby. Thus G is an argument graph. Further, since a is a tree it has a unique source, n (where $label(n) = claim(\mathsf{a})$). It is straightforward to show that $label(n)$ is the only source in G. Thus G is a focused argument graph with $claim(\mathsf{a}) = claim(G)$.

G clearly represents a. For define f so that, for each node n of a, $f(n)$ is $label(n)$ (this is well-defined and surjective w.r.t. $v(G)$). We need to check whether, if $f(n) = s$, then $labels(\{n' \mid n' \in children(n)\}) = \{s' \mid (s, s') \in e(G)\}$. Assume $f(n) = s$, so that $s = label(n)$. If $s' \in labels(\{n' \mid n' \in children(n)\}) \setminus \{\top\}$ then $(s, s') \in e(G)$ by definition of $e(G)$; if $s' \in \{s' \mid (s, s') \in e(G)\}$, then this can only be because there is n' in a such that $n \in children(n')$ and $label(n') = s'$, so that $s' \in labels(\{n^* \mid (n, n^*) \in edges(\mathsf{a})\}) \setminus \{\top\}$. Thus $labels(\{n^* \mid (n, n^*) \in edges(\mathsf{a})\}) = \{s^* \mid (s, s^*) \in e(G)\}$.

Since $v(G)$ is just $\{label(n) \mid n \in nodes(\mathsf{a})\}$, it is immediate that $support(\mathsf{a}) = support(G)$.

Plainly, G is a graphical conversion of a by construction. We must show that G is the unique graphical conversion of a. Thus suppose for contradiction that G' is some other graphical conversion of a. Then since $claim(G) = claim(G')$, it must be that there is $s \in v(G) \cap v(G')$ such that $\{s' \mid (s, s') \in e(G)\} \neq \{s' \mid (s, s') \in e(G')\}$. But then $\{\mathsf{a}\}$ is bloated, and so a is either circular or flabby. Contradiction.

For (ii), let G be

$$v(G) = \{s \mid \exists \mathsf{a} \in \mathsf{A}[s \in labels(nodes(\mathsf{a}))]\} \setminus \{\top\}$$

$$e(G) = \{(s, s') \mid s' \neq \top, \exists \mathsf{a} \in \mathsf{A} \exists n, n' \in nodes(\mathsf{a})[label(n) = s, label(n') = s', n' \in children(n)]\}$$

Similar reasoning to that in (i) shows that G is an argument graph. Plainly $claims(\mathsf{A}) \subseteq v(G)$, and since G is an argument graph we must have that for $a \in \mathcal{A} \cap v(G)$, then $a \in support(G)$; thus $\bigcup_{a\in\mathcal{A}} support(\mathsf{a}) = support(G)$. That G is a graphical conversion of A follows by construction of G, and the uniqueness of G follows by an argument similar to that for (i). So too for G's representing each argument in A. □

A.2. Theorem 4.7

Let $s \in \mathcal{L}$ and $A \in \mathcal{A}$. (i) For every non-circular, non-flabby tree-based argument for s supported by A there exists a tree-based argument for s supported by A. (ii) For every tree-based argument for s supported by A there exists a non-circular, non-flabby tree-based argument for s supported by $A' \subseteq A$.

Proof. For (i), non-circular, non-flabby arguments are, of course, arguments. For (ii), our strategy is to show how to transform any argument into a non-circular, non-flabby one.

Thus let a be an argument for s supported by A. Recall Algorithm 1 from [9], reprinted here as Algorithm A.1.[14] This takes an arbitrary tree-based argument a and 'reduces' it to an argument a′. At lines 6 and 9 the algorithm performs non-deterministic choices (of a node/sentence and of a subtree, respectively). By making alternative such choices different arguments can be obtained. For example, consider the application of the algorithm to the left-most argument in Fig. 28. Depending on the choice of subtree at line 9, the algorithm may return the middle or the right-most argument above.

We will say that, where a′ may be obtained from a using the algorithm with particular choices, then a′ *is a reduction of* a, and write this as $reduce(\mathsf{a}, \mathsf{a}')$. Thus, if $\textsc{reduce}(\mathsf{a}) = \mathsf{a}'$ for some particular choices, then $reduce(\mathsf{a}, \mathsf{a}')$ holds.

Lemma A.1. *If* $reduce(\mathsf{a}, \mathsf{a}')$, *then* a′ *is non-circular and non-flabby,* $claim(\mathsf{a}) = claim(\mathsf{a}')$ *and* $support(\mathsf{a}') \subseteq support(\mathsf{a})$.

[14] Here: $rank(N, T)$ returns the length of the path from N to the root of tree T; $rank(T)$ returns the maximum rank of any node in tree T; $path(N, N', T)$ returns the set of nodes on the (unique) path from N to N' (not including N) in tree T; $substitute(T, N, T')$ takes tree T and replaces the subtree rooted at N by tree T'; $nodes(T)$, $root(T)$ and $subTrees(T)$ return, respectively, the set of nodes, the root and the set of subtrees of tree T; $pickOne(S)$ chooses a member of S; $label(N, \mathsf{a})$ returns the label of node N in argument a (this is a member of $\mathcal{L}$ or $\top$, see §2).

Algorithm A.1 REDUCE(a: argument).

```
1: $r := 0$
2: $seen := \{\}$
3: while $r \leqslant rank(\mathrm{a})$ do
4:     $nodes := \{N \in nodes(\mathrm{a}) \mid (rank(N, \mathrm{a}) = r) \wedge (label(N, \mathrm{a}) \in \mathcal{L} \setminus \mathcal{A}) \wedge (N \notin seen)\}$
5:     while $nodes \neq \{\}$ do
6:         $N := pickOne(nodes)$
7:         $s := label(N, \mathrm{a})$
8:         $leafTrees := \{\mathrm{b} \in subTrees(\mathrm{a}) | (label(root(\mathrm{b})) = s) \wedge \neg\exists N'[N' \in nodes(\mathrm{b}) \setminus \{root(\mathrm{b})\} \wedge label(N', \mathrm{a}) = s]\}$
9:         $\mathrm{b} := pickOne(leafTrees)$
10:        for all $N' \in nodes(\mathrm{a})$ s.t. $label(N', \mathrm{a}) = s \wedge \neg\exists X[X \in path(N', root(\mathrm{a}), \mathrm{a}) \wedge label(X) = s]$ do
11:            $\mathrm{a} := substitute(\mathrm{a}, N', \mathrm{b})$
12:        end for
13:        $seen := seen \cup \{N \in nodes | label(N) = s\}$
14:        $nodes := nodes \setminus seen$
15:    end while
16:    $r := r + 1$
17: end while
18: return a
```

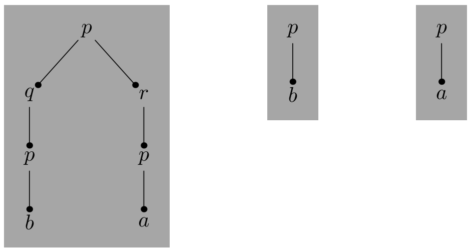


Fig. 28. If Algorithm A.1 is applied to the left-most argument, either of the other two may result.

Proof. First, Algorithm A.1 terminates, since (i) a is finite and thus $rank(\mathrm{a})$ is finite; (ii) there are finitely many nodes at lines 10, 13; (iii) at every iteration of the external while loop the set *nodes* is smaller. Secondly, a' is a subtree of a with the same root, and so the same claim as a, thus $support(\mathrm{a}') \subseteq support(\mathrm{a})$. Thirdly, trivially a' is a tree-based argument. Finally, by construction, each sentence in a' is proven by only one rule; thus $\{\mathrm{a}'\}$ cannot be bloated and so by Theorem 3.3, a' is non-circular and non-flabby. □

This completes the proof of Theorem 4.7: Algorithm A.1 can be used to find, for any argument a, some argument a' such that $reduce(\mathrm{a}, \mathrm{a}')$, and Lemma A.1 shows that such an a' will be non-circular and non-flabby. □

A.3. Theorem 4.16

Any argument graph characteristic function f_{R} has a least fixed point equal to $f^{\omega}_{\mathsf{R}}(\lozenge)$.

Proof. We use the Kleene fixed-point theorem. We must show that f_{R} is Scott-continuous (sometimes just called 'continuous').

First, some preliminary definitions.

Definition A.2. Let $G_1, \dots, G_n$ be argument graphs. They are said to be *compatible* (which we write as $G_1 \parallel \dots \parallel G_n$) if there is a maximally rule-consistent R such that $(rules(G_1) \cup \dots \cup rules(G_n)) \subseteq \mathsf{R}$.

$G_1 \cup \dots \cup G_n$ is defined to be the graph

$$(v(G_1) \cup \dots \cup v(G_n), e(G_1) \cup \dots \cup e(G_n))$$

and $G_1 \cap \dots \cap G_n$ is

$$(v(G_1) \cap \dots \cap v(G_n), e(G_1) \cap \dots \cap e(G_n)).$$

(Note that these are not, in general, argument graphs.) ⌟

Let us note that $G_1 \parallel \dots \parallel G_n$ iff $G_1 \cup \dots \cup G_n$ is an argument graph.

Lemma A.3. *Let R be maximally rule-consistent, and G_{R} the set of argument graphs G such that $rules(G) \subseteq \mathsf{R}$. Then $(\mathsf{G}_{\mathsf{R}}, \subseteq)$ is a complete lattice, such that for $X \subseteq \mathsf{G}_{\mathsf{R}}$, the least upper bound of X ($lub(X)$) is $\bigcup X$, and the greatest lower bound ($glb(X)$) is $\bigcap X$.*

Proof. Evidently $\subseteq$ partially orders G_R. Thus let $X \subseteq \mathsf{G}_\mathsf{R}$; note that X is finite. As $rules(G) \subseteq \mathsf{R}$ for any $G \in X$, then letting $X = \{G_1, \dots, G_n\}$, we have $G_1 \parallel \cdots \parallel G_n$. Then $\bigcup X$ is evidently an argument graph, and $(\bigcup X) \in \mathsf{G}_\mathsf{R}$. It is trivial to show that this is $lub(X)$. The result for $glb(X)$ follows similarly. □

The following lemma shows that f_R is Scott-continuous (sometimes called 'ω-continuous').

Lemma A.4. *Let f_R the argument graph characteristic function w.r.t. the maximally rule-consistent R. Then for any $X \subseteq \mathsf{G}_\mathsf{R}$ such that $lub(X) \in X$:*

$$f_\mathsf{R}(lub(X)) = lub(\{f_\mathsf{R}(G) \mid G \in X\}).$$

Proof. G_R must be finite, so let X be $\{G_1, \dots, G_n\}$. We need to show that $f_\mathsf{R}(lub(\{G_1, \dots, G_n\})) = lub(\{f_\mathsf{R}(G_1,), \dots, f_\mathsf{R}(G_n)\})$. Let $G_1 \cup \cdots \cup G_n$ be G^*. So, we need that $f_\mathsf{R}(G^*) = \bigcup\{f_\mathsf{R}(G_1), \dots, f_\mathsf{R}(G_n)\}$. Since $G_1 \subseteq G^*, \dots, G_n \subseteq G^*$, then by monotonicity of f_R, $f_\mathsf{R}(G_1) \subseteq f_\mathsf{R}(G^*), \dots, f_\mathsf{R}(G_n) \subseteq f_\mathsf{R}(G^*)$. Thus $\bigcup\{f_\mathsf{R}(G_1), \dots, f_\mathsf{R}(G_n)\} \subseteq f_\mathsf{R}(G^*)$. But since $lub(X) \in X$ just means that $G^* \in \{G_1, \dots, G_n\}$, it must be that $f_\mathsf{R}(G^*) \subseteq \bigcup\{f_\mathsf{R}(G_1), \dots, f_\mathsf{R}(G_n)\}$. □

Now, the Kleene fixed-point theorem can be applied to yield that $f_\mathsf{R}^\omega(\lozenge)$ is the least fixed point of f_R. □

A.4. Theorem 4.20

Let G be an argument graph and A_G the set of arguments represented in G.

i. If G is admissible, so is A_G.
ii. If G is complete, then there is a complete extension A^* such that $\mathsf{A}_G \subseteq \mathsf{A}^*$ and $claims(\mathsf{A}_G) = claims(\mathsf{A}^*)$.
iii. If G is grounded and A^* is the grounded extension, then $\mathsf{A}_G \subseteq \mathsf{A}^*$ and $claims(\mathsf{A}_G) = claims(\mathsf{A}^*)$.

Proof. We take the parts i–iii in turn. The proof of part i depends on the following lemma, which relates properties of argument graphs and reductions of arguments, as defined in the proof of Theorem 4.7 above.

Lemma A.5. *If* a *is a tree-based argument:*

I. *there is a focused argument graph G, with $claim(\mathsf{a}) = claim(G)$ and $support(G) \subseteq support(\mathsf{a})$;*
II. *if G is a graphical conversion of some a' such that $reduce(\mathsf{a}, \mathsf{a}')$, then G is a graphical conversion of* a *(cf. Definition 4.5 for the notion of graphical conversion).*

Proof.

I. Directly from Theorems 4.6 and 4.7.
II. To show that G is a conversion of a, we must further show that if $(s_1, s_2) \in e(G)$, then there is an edge $(n_1, n_2) \in \mathsf{a}$ such that $label(n_1) = s_1$ and $label(n_2) = s_2$. Suppose $(s_1, s_2) \in e(G)$. Then there must be nodes n_1, n_2 in the tree a' such that $label(n_1) = s_1$ and $label(n_2) = s_2$ and (n_1, n_2) is an edge in a. (Note that not all pairs of nodes in a' with these labels will be connected by an edge in a; but some such pair must exist.) □

We now prove the parts i–iii of Theorem 4.20 successively.

i. Assume G is admissible. Then it is conflict-free. If A_G is not conflict-free, then there are $\mathsf{a}, \mathsf{b} \in \mathsf{A}_G$ such that $\mathsf{a} \rightsquigarrow \mathsf{b}$ (a and b need not be distinct). Since $claim(\mathsf{a}) \in v(G)$ and $support(\mathsf{b}) \subseteq v(G)$, then $G \rightsquigarrow G$. Contradiction; so A_G is conflict-free. Let $\mathsf{a}' \notin \mathsf{A}_G$ be such that $\mathsf{a}' \rightsquigarrow \mathsf{A}_G$. Let s' be $claim(\mathsf{a}')$. Then by Lemma A.5 there is an argument graph G' such that $s' = claim(G')$ and $support(G') \subseteq support(\mathsf{a}')$. Since $\mathsf{a}' \rightsquigarrow \mathsf{A}_G$, there is $a \in support(\mathsf{A}_G)$ such that $\bar{a} = s'$. Evidently $a \in support(G)$, so that $G' \rightsquigarrow G$. Since G is admissible, then by Theorem 4.19.i, $G \rightsquigarrow G'$. Then there must be $s \in v(G)$ with $s = \bar{a'}$ for $a' \in v(G')$, and clearly $a' \in support(\mathsf{a}')$. But since $s \in v(G)$, then by Theorem 4.3 there is $\mathsf{a}^* \in \mathsf{A}_G$ with $claim(\mathsf{a}^*) = s$, and $\mathsf{a}^* \rightsquigarrow \mathsf{a}'$. Thus A_G defends itself against a'.
ii. Assume G is complete. We first show that $\mathsf{A}_G \cup args(support(G))$ is admissible. Suppose not; then it is either not conflict-free, or it does not defend itself. Plainly it must defend itself from attacks, for A_G is admissible and $support(\mathsf{A}_G) = support(\mathsf{A}_G \cup args(support(G)))$. Then $(\mathsf{A}_G \cup args(support(G))) \rightsquigarrow (\mathsf{A}_G \cup args(support(G)))$. Clearly $\mathsf{A}_G \not\rightsquigarrow (\mathsf{A}_G \cup args(support(G)))$. If $args(support(G)) \rightsquigarrow \mathsf{A}_G$ then since A_G is admissible, $\mathsf{A}_G \rightsquigarrow args(support(G))$. But then $\mathsf{A}_G \rightsquigarrow \mathsf{A}_G$—contradiction. If $args(support(G)) \rightsquigarrow args(support(G))$, then $args(support(G)) \rightsquigarrow \mathsf{A}_G$, which again leads to a contradiction. So $\mathsf{A}_G \cup args(support(G))$ is admissible.
We then show that $\mathsf{A}_G \cup args(support(G))$ contains all the arguments it can defend. First we prove that (*) if G is complete, then $claims(\mathsf{A}_G) = claims(\mathsf{A}_G \cup args(support(G)))$. The $\subseteq$ direction is trivial. The other direction follows by

noting that if a is such that $claims(subargs(\mathsf{a}) \setminus \{\mathsf{a}\}) \subseteq claims(\mathsf{A}_G)$ and $\mathsf{a} \in \mathsf{A}_G \cup args(support(G))$, then it follows by G being complete that $claim(\mathsf{a}) \in v(G)$, so that $claim(\mathsf{a}) \in claim(\mathsf{A}_G)$.

So, now suppose that $\mathsf{A}_G \cup args(support(G))$ does not contain all the arguments it defends. Let b be such an argument: $\mathsf{b} \notin \mathsf{A}_G \cup args(support(G))$, but for all c such that $\mathsf{c} \rightsquigarrow \mathsf{b}$, $(\mathsf{A}_G \cup args(support(G))) \rightsquigarrow \mathsf{c}$. Consider G^+, where $v(G^+) = v(G) \cup support(\mathsf{b})$, and $e(G^+) = e(G)$. Evidently $G^+ \neq G$, since as $\mathsf{b} \notin \mathsf{A}_G \cup args(support(G))$, $support(\mathsf{b}) \setminus v(G)$ is non-empty. G^+ is clearly an argument graph such that $G \subseteq G^+$; we will show that G defends it from attacks, contradicting (given Theorem 4.19.ii) that G is complete. Thus assume that $G' \rightsquigarrow G^+$ but $G' \not\rightsquigarrow G$ (if $G' \rightsquigarrow G$ then plainly $G \rightsquigarrow G'$ by admissibility of G). Let c^* be some argument represented by G' such that the $claim(\mathsf{c}^*) = \bar{b}$ for $b \in support(\mathsf{b}) \setminus v(G)$. Since $\mathsf{c}^* \rightsquigarrow \mathsf{b}$, $(\mathsf{A}_G \cup args(support(G))) \rightsquigarrow \mathsf{c}^*$. Thus there is some $\mathsf{a}' \in (\mathsf{A}_G \cup args(support(G)))$ such that $claim(\mathsf{a}') = \bar{c}$ for some $c \in support(\mathsf{c}^*)$. But by our result (*) above, $claim(\mathsf{a}') \in claims(\mathsf{A}_G)$, so that $claim(\mathsf{a}') \in v(G)$. So $G \rightsquigarrow G'$, and so G defends G^+. So, from Theorem 4.19.ii, G cannot be complete: contradiction.

Thus $\mathsf{A}_G \cup args(support(G))$ must contain all the arguments it defends, and so is complete.

iii. Suppose G is grounded. Then it is complete, and so, by part ii, $\mathsf{A}_G \subseteq A^c$ for some complete extension A^c. To show that $\mathsf{A}_G \subseteq \mathsf{A}^*$ for A^* grounded we must show that there is no complete A' such that $\mathsf{A}' \subset \mathsf{A}_G$. Assume there is such an A'; then it must be that $\mathsf{A}^* \subset \mathsf{A}_G$. Let G^* be a conversion of A^*. By Theorem 4.21.ii, G^* is complete. Clearly $G^* \subset G$, so that G is not $\subseteq$-minimally complete. Thus, given Theorem 4.19.iii, G cannot be grounded. Contradiction.

Since $\mathsf{A}_G \subseteq \mathsf{A}^*$, we have $claims(\mathsf{A}_G) \subseteq claims(\mathsf{A}^*)$. We must show $claims(\mathsf{A}^*) \subseteq claims(\mathsf{A}_G)$. Assume $s \in claims(\mathsf{A}^*)$. Since G is grounded, it is complete, so, by part ii, there is some complete A^c such that $claims(\mathsf{A}_G) = claims(\mathsf{A}^c)$. But since $s \in claims(\mathsf{A}^*)$, $s \in claims(\mathsf{A}^c)$. So $s \in claims(\mathsf{A}_G)$. □

A.5. Theorem 4.21

Let A be a set of tree-based arguments.

i. If A is admissible, then for all conversions G of A, G is admissible.
ii. If A is complete, then for all conversions G of A, G is complete.
iii. If A is grounded, then there exists a conversion G of A such that G is grounded.

Proof.

i. Assume A is admissible, but that, for some conversion G of A, G is not admissible, for contradiction. If $G \rightsquigarrow G$, then there are $s, a \in v(G)$ such that $\bar{a} = s$. Since G is a conversion of A, it must be that $v(G) \subseteq labels(nodes(\mathsf{A}))$. Thus there is some $\mathsf{a} \in \mathsf{A}$ such that $a \in support(\mathsf{a})$. If there is an $\mathsf{a}' \in \mathsf{A}$ such that $s = claim(\mathsf{a}')$, then $\mathsf{A} \rightsquigarrow \mathsf{A}$: contradiction. Suppose there is no such a'. Then there must be $\mathsf{a}' \in \mathsf{A}$ such that $s \in labels(nodes(\mathsf{a}'))$. Let a'' be a subtree of a' such that $claim(\mathsf{a}'') = s$. Then $\mathsf{a}'' \rightsquigarrow \mathsf{A}$, so that $\mathsf{A} \rightsquigarrow \mathsf{a}''$. But evidently $support(\mathsf{a}'') \subseteq support(\mathsf{a}')$ so that $\mathsf{A} \rightsquigarrow \mathsf{A}$. So A cannot be admissible: contradiction. Thus $G \not\rightsquigarrow G$. Suppose that there is G' such that $G' \rightsquigarrow G$, and let a be an argument represented in G' such that $claim(\mathsf{a}) = \bar{a}$ for some $a \in support(G)$. Then evidently $\mathsf{a} \rightsquigarrow \mathsf{A}$, so that $\mathsf{A} \rightsquigarrow \mathsf{a}$. But then it follows that $G \rightsquigarrow G'$. Thus from Theorem 4.19.i, G must be admissible.
ii. Assume A is complete; then it is admissible, and so G is admissible by (i) of the present theorem. In light of Theorem 4.19.ii, we must show that there is no G' with $G \subset G'$, such that for all G^* where $G^* \rightsquigarrow G'$, $G \rightsquigarrow G^*$. Thus assume that there is such an argument graph G'. Let a be an argument represented in G' but not in G (since $G \subset G'$ there must be such an argument). Suppose $\mathsf{b} \rightsquigarrow \mathsf{a}$, and let G^{b} be a conversion of b; it must then be that $G^{\mathsf{b}} \rightsquigarrow G'$. So by assumption, $G \rightsquigarrow G^{\mathsf{b}}$. But this means that $\mathsf{A} \rightsquigarrow \mathsf{b}$, so that A defends a. But then A cannot be complete—contradiction. So there is no such G', and G is complete.
iii. Assume A is grounded. Thus A is the least fixed point of the characteristic function f (see § 2), i.e., A=$f^{\omega}(\emptyset)$. For each $s \in claims(\mathsf{A})$, let the *least rank* of s be the least $n \in \mathbb{N}$ such that there is $\mathsf{a} \in f^n(\emptyset)$ with $claim(\mathsf{a}) = s$; this is evidently well-defined. Let R be a *rule of least rank for* s if there is $\mathsf{a} \in f^n(\emptyset)$ with $s = claim(\mathsf{a})$, such that $R = last(\mathsf{a})$[15] and n is the least rank of s. Let $\mathsf{R}' \subseteq \mathcal{R}$ be *minimally grounding* if for each $s \in claims(\mathsf{A})$, R' contains precisely one rule of least rank for s and no other rules. Let $\mathsf{R} \subseteq \mathcal{R}$ be *grounding* if R is maximally rule-consistent and $\mathsf{R}' \subseteq \mathsf{R}$ for some minimally grounded R'. Plainly a grounding R always exists. We claim that $f^{\omega}_{\mathsf{R}}(\Diamond)$ is a conversion of A and is grounded. We first prove the following lemma.

Lemma A.6. *For all $n \in \mathbb{N}$, (I) $claims(f^n(\emptyset)) = v(f^n_{\mathsf{R}}(\Diamond))$ and (II) each argument represented in $f^n_{\mathsf{R}}(\Diamond)$ is a member of $f^n(\emptyset)$.*

Proof. By induction on n.

Base case. Trivial for $n = 0$ for both (I) and (II).

Induction step. Assume (I) and (II) are true in the case of $n = i$. We will show they both hold for $n = i + 1$.

[15] Here, $last(\mathsf{a})$ stands for the rule $(s \leftarrow s_1, \ldots, s_m) \in \mathcal{R}$ such that $s_1, \ldots, s_m$ are the children of s in a, if $m \geqslant 1$, and $\top$ is the only child of s in a, if $m = 0$, where $s = claim(\mathsf{a})$.

I) By inductive hypothesis $claims(f^i(\emptyset)) = v(f^i_{\mathsf{R}}(\Diamond))$.
Assume $s \in claims(f^{i+1}(\emptyset))$. If $s \in claims(f^i(\emptyset))$ we are done, so suppose not. Let G^s be the focused argument graph whose claim is s, such that $rules(G^s) \subseteq \mathsf{R}$ (this can easily be shown to exist), and let G^+ be $G^s \cup f^i_{\mathsf{R}}(\Diamond)$; G^+ is easily shown to be an argument graph. Suppose that $G^* \rightsquigarrow G^+$. If (*) $G^* \rightsquigarrow f^i_{\mathsf{R}}(\Diamond)$, then $f^i_{\mathsf{R}}(\Diamond) \rightsquigarrow G^*$. If (**) $G^* \rightsquigarrow G^s$, then the (unique) argument b represented by G^* whose claim is $claim(G^*)$ is such that $\mathsf{b} \rightsquigarrow \mathsf{a}$ where a is the unique argument represented by G^s whose claim is s. Then $\mathsf{b} \rightsquigarrow f^{i+1}(\emptyset)$, so that $f^i(\emptyset) \rightsquigarrow \mathsf{b}$, which by the inductive hypothesis means that $f^i_{\mathsf{R}}(\Diamond) \rightsquigarrow G^*$. Therefore in both (*) and (**), $f^i_{\mathsf{R}}(\Diamond) \rightsquigarrow G^*$, So $G^s \subseteq f^{i+1}(\Diamond)$, and thus $s \in v(f^{i+1}_{\mathsf{R}}(\Diamond))$.
Assume now that $s \in v(f^{i+1}_{\mathsf{R}}(\Diamond))$, and let a be the (unique) argument represented in $f^{i+1}_{\mathsf{R}}(\Diamond)$ such that $claim(\mathsf{a}) = s$. Since R is grounding, it must be that $\mathsf{a} \in f^{i+1}(\emptyset)$, so that $s \in claims(f^{i+1}(\emptyset))$.
II) That this is true is shown by the argumentation for the second half of part I), above.

By induction, we conclude our result. □

Now, where R is grounding, then $f^{\omega}_{\mathsf{R}}(\Diamond)$ must plainly be a conversion of A. We must show that $f^{\omega}_{\mathsf{R}}(\Diamond)$ is grounded. It is plainly conflict-free, given that A is. Take any maximally rule-consistent R' such that $rules(f^{\omega}_{\mathsf{R}}(\Diamond)) \subseteq \mathsf{R}'$. Each such R' must be a grounding, and it is plain that $f^n_{\mathsf{R}'}(\Diamond) = f^n_{\mathsf{R}}(\Diamond)$ for all $n \in \mathbb{N}$. Since $f^{\omega}(\emptyset)$ is the least fixed point of f, $f^{\omega}_{\mathsf{R}}(\Diamond)$ is the least fixed point of f_{R}, and so too of $f^n_{\mathsf{R}'}$. Thus by Definition 4.17, $f^{\omega}_{\mathsf{R}}(\Diamond)$ is grounded. □

A.6. Theorem 5.6

Let $((\mathcal{P}_i, \boldsymbol{O}_i, \mathcal{G}_i, D_i, C_i))^n_{i=0}$ be an X-graph-DS for s_0. Then for all i such that $0 \leqslant i \leqslant n$ (if n is finite), or all i such that $0 \leqslant i < n$ (otherwise—i.e., if $n = \omega$):

i. $\mathcal{P}_i$ is a potential argument graph, and $s_0 \in v(\mathcal{P}_i)$;
ii. $\boldsymbol{O}_i$ is an argument graph set;
iii. $\mathcal{G}_i$ is a directed graph over $\mathcal{L}$;
iv. $D_i \subseteq \mathcal{A}$ and $C_i \subseteq \mathcal{A}$. ⌟

Proof. The proof proceeds by induction on the length $n \geqslant 0$ of the X-graph-DS. The base case is trivially true given the constraints on the tuple $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0)$ in Definition 5.5.

Assume the result holds for $n = k$. We will show it holds for $n = k + 1$. Thus let

$$(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_k, \boldsymbol{O}_k, \mathcal{G}_k, D_k, C_k), (\mathcal{P}_{k+1}, \boldsymbol{O}_{k+1}, \mathcal{G}_{k+1}, D_{k+1}, C_{k+1})$$

be an X-graph-DS for $s_0 \in \mathcal{L}$.

We must show that for all j, $0 \leqslant j \leqslant k + 1$, the components of the tuples fulfil the properties set in the theorem. Plainly $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_k, \boldsymbol{O}_k, \mathcal{G}_k, D_k, C_k)$ is a k-length X-graph-DS, so all the properties we want to prove hold for all tuples up to and including the k-th, by the induction hypothesis. Moreover, if any component in the $(k + 1)$-th tuple is the same as in the k-th, the result trivially holds again by the induction hypothesis. For the remaining case, we take the parts of the theorem in turn.

i. We need to prove that $\mathcal{P}_{k+1} \neq \mathcal{P}_k$ is a potential argument graph with $s_0 \in v(\mathcal{P}_{k+1})$.
Examination of Definition 5.5 shows that $\mathcal{P}_k$ is expanded to $\mathcal{P}_{k+1}$ according to case 1(ii) or case 2(i)(c).
In case 1(ii), plainly $\mathcal{P}_{k+1}$ is acyclic because of the acyclicity check in case 1(ii), and $v(\mathcal{P}_{k+1}) \subseteq \mathcal{L}$. $u(\mathcal{P}_k) \subseteq sinks(\mathcal{P}_k)$ by the induction hypothesis, and given the definition of $updtgrph$ from Definition 5.4, it must be that $u(\mathcal{P}_{k+1}) \subseteq sinks(\mathcal{P}_{k+1})$. Recall conditions (i) and (ii) from Definition 5.1. Plainly, no edge from an assumption is added, so that (i) remains satisfied. So assume that some $s' \in v(\mathcal{P}_{k+1})$ is such that $s' \notin \mathcal{A}$. If $s' \in u(\mathcal{P}_{k+1})$, we are done; so suppose instead that $s' \notin u(\mathcal{P}_{k+1})$. Then either s' was selected for the step to $\mathcal{P}_{k+1}$, in which case the result follows by the nature of case 1(ii); or else $s' \in (v(\mathcal{P}_k) \setminus u(\mathcal{P}_k))$ and the result follows by induction hypothesis. Thus Definition 5.1 is satisfied.
In case 2(i)(c), $\mathcal{P}_k \neq \mathcal{P}_{k+1}$ only if $\mathcal{P}_{k+1}$ is the same as $\mathcal{P}_k$ but with a single new (unmarked) node added, and no new edges. Plainly this means that Definition 5.1 is satisfied.
Thus, $\mathcal{P}_{k+1}$ is a potential argument graph. Evidently, since $s_0 \in v(\mathcal{P}_k)$ by the induction hypothesis and members of $v(\mathcal{P}_k)$ are not removed, $s_0 \in v(\mathcal{P}_{k+1})$.
ii. We must show that $\boldsymbol{O}_{k+1}$ is an argument graph set. By Definition 5.5, $\boldsymbol{O}_{k+1} \neq \boldsymbol{O}_k$ only if $\boldsymbol{O}_k$ is extended to $\boldsymbol{O}_{k+1}$ according to case 1(i), case 2(i)(a), case 2(i)(b), case 2(i)(c) or case 2(ii). Cases 1(i), 2(i)(a), 2(i)(b) and 2(i)(c) follow easily from Definition 5.4 and Definition 5.5. For case 2(ii), we must show that each member of (for the selected G and s):

$$\{updtgrph(G, s \leftarrow R, C_k) \mid R \in \mathsf{R}_C\} \cup \{updtgrph(G, s \leftarrow R, \emptyset) \mid R \in \mathsf{R}_{\neg C}\}$$

is a potential argument graph. This follows similarly to part (i) of the current theorem, above.

iii. We must show that, for $\mathcal{G}_{k+1} \neq \mathcal{G}_k$, $\mathcal{G}_{k+1}$ is a directed graph. Examination of Definition 5.5 shows that $\mathcal{G}_k$ is extended to $\mathcal{G}_k \cup_g S$, where S is a set of pairs of the form (s_1, s_2), for $s_1, s_2 \in \mathcal{L}$. With $X = \text{ADM}$, $\mathcal{G}_k \cup_g S$ by definition of $\cup_g$. Therefore, we only need to consider $X = \text{GRN}$. Here, the definition of $\cup_g$ makes it clear that $\mathcal{G}_{k+1}$ must be a directed graph over $\mathcal{L}$.
iv. Finally, the result for D_{k+1} and C_{k+1} is trivial given Definition 5.5.

Thus, for each part, the result holds for $n = k + 1$: this concludes the proof by induction. □

A.7. Theorem 5.9

For any X-graph-DD ($X \in \{\text{ADM}, \text{GRN}\}$) for s_0 with resulting argument graph $\mathcal{P}$, $\mathcal{P}$ is admissible and $s_0 \in v(\mathcal{P})$.

Proof. That $s_0 \in v(\mathcal{P})$ follows directly from Theorem 5.6. To prove the rest, let

$$(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$$

be the given X-graph-DD, with $n \geqslant 0$, with $\mathcal{P}$ corresponding to $\mathcal{P}_n$. We will make use of the two lemmas below. Intuitively, the significance of the first lemma is that any attacker of the PROPONENT is an expansion of some potential argument in $\boldsymbol{O}_n$.

Lemma A.7. *If $a \in D_n$, then any focused argument graph G such that $\bar{a} = claim(G)$ is an expansion of some $G^* \in \boldsymbol{O}_n$.*

Proof. Suppose $a \in D_n$, and let G be a focused argument graph such that $\bar{a} = claim(G)$. Since $a \in D_n$, then from case 1(i) of Definition 5.5 we know that the focused potential argument graph $G_{\bar{a}} = (\{\bar{a}\}, \emptyset)$ will have been added to some $\boldsymbol{O}_i$. G is necessarily an expansion of $G_{\bar{a}}$. We prove that no step from $\boldsymbol{O}_j$ to $\boldsymbol{O}_{j+1}$, for $j \geqslant i$, removes any argument graphs for $\bar{a}$ which are expansions of members of $\boldsymbol{O}_j$.

Thus suppose G^+ is an expansion of some $G^- \in \boldsymbol{O}_j$. If the move from $\boldsymbol{O}_j$ to $\boldsymbol{O}_{j+1}$ is by case 2(i) of Definition 5.5, plainly G^+ would still be an expansion of some member (since G^- would be retained, but possibly differently marked). If the move is by case 2(ii), then if $s \notin \mathcal{A}$ from $G^- \in \boldsymbol{O}_j$ was chosen, it must be that there were no edges $(s, s') \in v(G^-)$. But then since G^+ is an expansion of G^-, there must be edges $(s, s_1), \ldots, (s, s_n)$ in that expansion, such that $s \leftarrow s_1, \ldots, s_n$ is a rule in $\mathcal{R}$. But then since for all rules of the form $s \leftarrow R$ in $\mathcal{R}$, there is $G_R \in \boldsymbol{O}_{j+1}$ such that

$$v(G_R) = v(G^-) \cup R,$$
$$e(G_R) = e(G^-) \cup \{(s, s') \mid s' \in R\}$$

(by case 2(ii) of Definition 5.5), then there must be some member of $\boldsymbol{O}_{j+1}$ of which G^+ is an expansion. Thus G, in particular, must be an expansion of some $G^* \in \boldsymbol{O}_n$. □

The second lemma concerns the existence of counter-attacks by the PROPONENT on expansions of OPPONENT argument graphs.

Lemma A.8. *If G is an expansion of a member of $\boldsymbol{O}_n$, then $\mathcal{P} \rightsquigarrow G$.*

Proof. Let G be an expansion of $G^- \in \boldsymbol{O}_n$. $G^- \notin u(\boldsymbol{O}_n)$ (since $u(\boldsymbol{O}_n) = \emptyset$) and thus will have been marked in the transition from some $\boldsymbol{O}_i$ to $\boldsymbol{O}_{i+1}$, either in the application, in Definition 5.5, of case 2(i)(b), case 2(i)(c), or in case 2(ii) as some $updtgrph(G^-, s' \leftarrow R, C_i)$ such that $R \in \mathsf{R}_C$, i.e., $R \cap C_i \neq \emptyset$.

If the case was 2(i)(b), then there is some $s \in v(G^-)$ such that $s \in C_i$. If the case was 2(i)(c), then there is some $s \in v(G^-)$ such that $s \in C_{i+1}$. If the case was 2(ii) then there must be some $s \in R$, and thus $s \in v(updtgrph(G^-, s' \leftarrow R, C_i))$, such that $s \in C_i$. So in all cases there is $s \in v(G^-)$ such that $s \in C_n$.

But examination of the cases of Definition 5.5 shows that for any s added to some C_j (only case 2(i)(c)), then there must be $\bar{s} \in v(\mathcal{P}_{i+1})$. It is easy to see that $v(\mathcal{P}_k) \subseteq v(\mathcal{P}_{k+1})$ for all $0 \leqslant k \leqslant n$. Then $\bar{s} \in v(\mathcal{P})$, and thus $\mathcal{P} \rightsquigarrow G$. □

Now we move to the proof of the theorem. Given Theorem 4.19(i), we must show that (I) $\mathcal{P} \not\rightsquigarrow \mathcal{P}$, and that (II) for any argument graph G such that $G \rightsquigarrow \mathcal{P}$, then $\mathcal{P} \rightsquigarrow G$.

I. First assume for contradiction that $\mathcal{P} \rightsquigarrow \mathcal{P}$. Then there must be $s, a \in v(\mathcal{P})$ such that $\bar{a} = s$. Let G_s be the focused argument graph such that $G_s \subseteq \mathcal{P}$ and $claim(G_s) = s$. Since $\bar{a} = s$, by Definition 5.5 case 1(i) the focused potential argument graph $(\{s\}, \emptyset)$ was added to some $\boldsymbol{O}_i$. In this case $support(\mathcal{P}) = D_n$. Then, by Lemma A.7, any expansion of a member of $\boldsymbol{O}_i$ must also be an expansion of a member of $\boldsymbol{O}_n$; G_s is such an expansion, and let G_s^- be the member of $\boldsymbol{O}_n$ of which it is an expansion. Since $G_s^- \notin u(\boldsymbol{O}_n)$ (because $u(\boldsymbol{O}_n) = \emptyset$), G_s^- must have been marked by case 2(i)(b) or 2(i)(c) of Definition 5.5. But then there is some $b \in support(G_s^-)$ such that $b \in C_n$, and clearly $b \in support(G_s)$, too. Evidently also $\mathcal{P} \rightsquigarrow G_s$.

Now, $G_s \subseteq \mathcal{P}$, so clearly $support(G_s) \subseteq v(\mathcal{P})$. If $s' \in \mathcal{L}$ is added to some $\mathcal{P}_i$, it is clear from Definition 5.5 (cases 1(ii), 2(i)(c)) and from Definition 5.7 that if $s' \in \mathcal{A}$, then s' is simultaneously added to D_i. Thus $support(G_s) \subseteq D_n$, and so $b \in D_n \cap C_n$.
Either (A) b was added to D_n first; (B) b was added to C_n first; or (C) b was added to D_n and C_n at the same time.
If (A), then b is in some D_i and was added as a culprit into C_{i+1} by case 2(i)(c). This is impossible, as the preconditions of that case state that $b \notin D_i$. If (B), then b is in some C_i and was added as defence to D_{i+1}, by case 1(ii) or case 2(i)(c). Here, again, the preconditions of both case 1(ii) and case 2(i)(c) mean this is impossible. Finally, if (C), then it must be that the case is 2(i)(c), with some $G^* \in \boldsymbol{O}_i$, and $b \in G^*$, $\bar{b} = b$, with $b \in \mathcal{A}$. b has not already been added by case 1(ii) to D_i. Then since $\bar{b} = b$, b cannot be the claim of G^*. Case 2(i)(c) adds b as unmarked to $\mathcal{P}_{i+1}$, and since $b \in \mathcal{A}$ then for some later $j > i$, case 1(i) must add $(\{b\}, \emptyset)$ to $\boldsymbol{O}_{j+1}$. Yet then there will be no sub-case of case (2)(i) which can apply: cases 2(i)(b) and 2(i)(c) fail on the precondition that $b \notin D_i$, and if b is ignored, i.e. case 2(i)(a), then $(\{b\}, \emptyset)$, with b unmarked, will be replaced by $(\{b\}, \emptyset)$, with b marked, which is impossible (as the resulting unmarked potential argument graph cannot be eliminated from $\mathcal{O}_{i+1}$ and thus $u(\mathcal{O}_n)$ cannot be empty). Thus none of (A), (B) or (C) is possible, and so $\mathcal{P} \not\rightsquigarrow \mathcal{P}$.

II. Assume $G \rightsquigarrow \mathcal{P}$. Then by Lemma A.7, G is the expansion of some $G^- \in \boldsymbol{O}_n$. By Lemma A.8, this means that $\mathcal{P} \rightsquigarrow G$. □

A.8. Theorem 5.10

For any GRN-graph-DD with resulting argument graph $\mathcal{P}$, there is some grounded argument graph G such that $\mathcal{P} \subseteq G$.

Proof. Our strategy will be as follows. First, we define notions of 'argument graph dispute tree' (Definition A.9) and 'corresponding argument graph dispute tree' (Definition A.10), which we show to be finite (Lemma A.11). Secondly, we prove a series of lemmas concerning relations between argument graphs in general. Thirdly, we present the proof itself, the core of which is by induction.

Definition A.9. An *argument graph dispute tree* $\mathcal{T}$ for $s \in \mathcal{L}$ is defined as follows:

i. Every node of $\mathcal{T}$ is a PROPONENT node or an OPPONENT node, but not both. The status of a child node is different from that of its parent. Each node is labelled by a focused argument graph.
ii. The root is a PROPONENT node labelled by an argument graph whose claim is s.
iii. For every PROPONENT node n labelled by G, and for every focused argument graph G' such that there is $a \in support(G)$ with $\bar{a} = claim(G')$, there is an OPPONENT child n' of n labelled by G' (thus, $G' \rightsquigarrow G$).
iv. For every OPPONENT node n labelled by G, there exists exactly one (PROPONENT) child n' of n, labelled by a focused argument graph G' such that there is some $a \in support(G)$ and $\bar{a} = claim(G')$ (thus, $G' \rightsquigarrow G$).
v. For any argument graphs G_1 and G_2 labelling any two PROPONENT nodes, there is some argument graph G such that $G_1 \subseteq G$ and $G_2 \subseteq G$.
vi. There are no other nodes except those given by (i)–(v). ⌟

This definition is intended to mirror the definition of dispute tree in [12], given later as Definition A.20 in the proof of Theorem A.10. Note that point (v) in the above constrains the argument graphs labelling the PROPONENT nodes to be 'compatible', in the sense of Definition A.2. This ensures that the argument graphs of the PROPONENT nodes can be 'gathered together' to form a single argument graph.

Where G is an argument graph and $s \in v(G)$, we will let $prune(s, G)$ denote the $\subseteq$-largest focused argument graph G' such that $G' \subseteq G$ and $claim(G') = s$. (This is guaranteed to exist and be unique, as is easy to show.)

Definition A.10. $\mathcal{T}$ is defined to be a *corresponding argument graph dispute tree* for a GRN-graph-DD with resulting $\mathcal{P}$ iff:

i. The root of $\mathcal{T}$ is a PROPONENT node labelled by $prune(s_0, \mathcal{P})$.
ii. For every PROPONENT node n labelled by G such that $a \in support(G)$, and for every focused argument graph G' such that $claim(G') = \bar{a}$, there is an OPPONENT child of n labelled by G'.
iii. For every OPPONENT node n labelled by G, there exists a single PROPONENT child n' of n, such that there is $a \in support(G)$ and n' is labelled by $prune(\bar{a}, \mathcal{P})$.
iv. There are no other nodes except those given by (i)–(iii). ⌟

We must of course show that a corresponding argument graph dispute tree according to Definition A.10, is an argument graph dispute tree according to Definition A.9. That, and the finiteness of this argument graph dispute tree, are shown as follows.

Lemma A.11. *Let $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$ be a GRN-graph-DD for $s_0 \in \mathcal{L}$ and $\mathcal{T}$ a corresponding argument graph dispute tree. Then $\mathcal{T}$ is an argument graph dispute tree in the sense of Definition A.9 and $\mathcal{T}$ is finite.*

Proof. Condition (ii) of Definition A.9 is met, since $s_0 \in \mathcal{P}_n$ and thus $prune(s_0, \mathcal{P}_n)$ is a focused argument graph. By Lemmas A.7 and A.8 every OPPONENT node is attacked by some PROPONENT node labelled by a focused argument graph G s.t. $G \subseteq \mathcal{P}_n$. Thus condition (iv) is met. Conditions (i) and (iii) are met trivially. Condition (v) is ensured, since the argument graphs labelling the PROPONENT edges have the form $prune(s', \mathcal{P}_n)$ for $s' \in \mathcal{P}_n$, and $\mathcal{P}_n$ is an argument graph.

Assume for contradiction that $\mathcal{T}$ is infinite. It cannot be infinitely branching (i.e., it cannot be that some node in $\mathcal{T}$ has infinitely many children), for this would only happen if there were infinitely many OPPONENT children of some PROPONENT node; but given the finiteness of $\mathcal{L}$ (and thus $\mathcal{R}$), then there are only finitely many focused argument graphs having a given claim. Therefore there must be an infinite path in $\mathcal{T}$. That implies, however, that there would be a cycle in $\mathcal{G}_n$, and this is impossible. So $\mathcal{T}$ is finite. □

This concludes the preliminaries on argument graph dispute trees.

We move now to the second part, and prove a series of lemmas about argument graphs. Some involve the following notions.

Definition A.12. Where G_1 and G_2 are argument graphs, then $G_1 \cup G_2$ is defined as $(v(G_1) \cup v(G_2), e(G_1) \cup e(G_2))$. (Note that, in general, this need not be an argument graph.)

Where G is an argument graph, then G' is said to be *acceptable w.r.t.* G iff $G \parallel G'$ and for all G'' such that $G'' \rightsquigarrow G'$, then $G \rightsquigarrow G''$. ⌟

The next lemma is a straightforward result on admissibility.

Lemma A.13. *Let G be an admissible argument graph and let G' be an argument graph such that $G \parallel G'$ and G' is acceptable with respect to G. Then $G \cup G'$ is admissible.*

Proof. We first show $G \cup G'$ is conflict-free. Clearly $G \not\rightsquigarrow G$. If $G \rightsquigarrow G'$ then by acceptability of G' w.r.t. G, $G \rightsquigarrow G$, which is a contradiction—so $G \not\rightsquigarrow G'$. If $G' \rightsquigarrow G'$, then $G \rightsquigarrow G'$ by acceptability, whence a contradiction arises as before. So $G' \not\rightsquigarrow G'$. If $G' \rightsquigarrow G$ then $G \rightsquigarrow G'$ by admissibility of G, whence the same contradiction. So $G \cup G'$ is conflict-free.

Now suppose some argument graph G'' is such that $G'' \rightsquigarrow (G \cup G')$. Evidently $G'' \rightsquigarrow G$ or $G'' \rightsquigarrow G'$. In each case $G \rightsquigarrow G''$, so $(G \cup G') \rightsquigarrow G''$. □

The next lemma shows that admissible argument graphs which are subgraphs of a grounded argument graph can be extended by argument graphs which are acceptable with respect to them; the result is still a subgraph of a grounded argument graph.

Lemma A.14. *Let G^- be an admissible argument graph such that $G^- \subseteq G'$ for some grounded argument graph G'. Let G be an argument graph such that $G^- \parallel G$ and $G \nsubseteq G^-$. If G is acceptable w.r.t. G^-, then $(G^- \cup G) \subseteq G^*$ for some grounded argument graph G^*.*

Proof. Since G' is grounded, then by Theorem 4.19(iii) there is no complete argument graph G'' such that $G'' \subset G'$. Since $G^- \subseteq G'$, this means there is no complete argument graph G'' such that $G'' \subset G^-$. Further, G^- is not itself complete, for G is acceptable w.r.t. G^- and $G^- \neq (G^- \cup G)$ (because $G \nsubseteq G^-$). Any G^+ such that $G^- \subseteq G^+ \subset (G^- \cup G)$ will not be complete, since clearly such a $(G^- \cup G)$ is acceptable w.r.t. such a G^+ (and strictly contains it). Thus there is no complete G'' such that $G'' \subset (G^- \cup G)$.

Since G is acceptable w.r.t. G^-, then for some f_{R} such that $rules(G^-) \subseteq \mathsf{R}$, we have that $(G^- \cup G) \subseteq f_{\mathsf{R}}(G^-)$. Thus since there is no complete subset of $(G^- \cup G)$, the least fixed point, G^{R}, of f_{R} must be such that $(G^- \cup G) \subseteq G^{\mathsf{R}}$. But for any f_{R}, the least fixed point of f_{R} must be included in some grounded extension. Thus there is a grounded G^* such that $(G^- \cup G) \subseteq G^*$. □

Our final preliminary lemma shows that combining compatible argument graphs which are subgraphs of grounded argument graphs produces a result which is itself a subgraph of a grounded argument graph.

Lemma A.15. *Let $\mathcal{T}$ be a corresponding argument graph dispute tree for a* GRN*-graph-DD with resulting $\mathcal{P}$, and let $\mathcal{T}_1, \ldots, \mathcal{T}_n$ be the argument graphs rooted at the children of the children of the root of $\mathcal{T}$. For each $\mathcal{T}_i$ amongst the $\mathcal{T}_1, \ldots, \mathcal{T}_n$, let G_i be union of the argument graphs labelling the proponent nodes of $\mathcal{T}_i$. If there are grounded argument graphs $G_1^+, \ldots, G_n^+$ such that $G_1 \subseteq G_1^+, \ldots, G_n \subseteq G_n^+$, then there is a grounded argument graph G^+ such that $(G_1 \cup \cdots \cup G_n) \subseteq G^+$.*

Proof. We prove the result for $n = 2$; the extension to larger n follows easily.

Evidently $G_1 \parallel G_2$ by construction, since the argument graphs labelling PROPONENT nodes must be compatible. Let $G_1 \subseteq G_1^+$ and $G_2 \subset G_2^+$ as stipulated. Since G_1^+ and G_2^+ are grounded, then where $\mathsf{A}_{G_1^+}$ is the set of arguments represented in G_1^+

and $\mathsf{A}_{G_2^+}$ is the set of arguments represented in G_2^+, Theorem 4.20(iii) requires that $claims(\mathsf{A}_{G_1^+}) = claims(\mathsf{A}_{G_2^+})$; Theorem 4.3 then requires that $v(G_1^+) = v(G_2^+)$. Let G^+ be defined such that $v(G^+) = v(G_1^+)$ and

$$e(G^+) = e(G_1) \cup e(G_2) \cup \{(s, s') \in e(G_1^+) \mid s \notin (v(G_1) \cup v(G_2))\}$$

Thus G^+ contains precisely the sentences in $v(G_1^+)$—which are precisely the sentences in $v(G_2^+)$—as well as all edges in G_1 or G_2, but any s which is not in $v(G_1)$ or $v(G_2)$ has children the same as the children of s in G_1^+. (Thus, G^+ can be thought of as matching the intersection of G_1 and G_2, and then being extended outside that intersection in ways that match G_1^+.)

We must show that G^+ is an argument graph. Suppose $s \in v(G^+)$. Then if $s \in \mathcal{A}$, there is plainly no edge of the form (s, s'). Suppose $s \notin \mathcal{A}$; we must show condition (ii) of Definition 4.1 is satisfied. If $s \in v(G_1)$ then there is a rule $s \leftarrow R$ such that for all $s' \in R$, there is an edge $(s, s') \in G^+$; there are plainly no other edges of the form $(s, s') \in G^+$, so that condition (ii) is satisfied. If $s \in v(G_2)$ a similar argument shows that condition (ii) is satisfied. If $s \notin (v(G_1) \cup v(G_2))$ then the definition of $e(G^+)$ above shows that (ii) is satisfied. Thus G^+ is an argument graph.

We now show that G^+ is grounded. First, we show that it is admissible. If G^+ were self-attacking, then since $v(G^+) = v(G_1^+) = v(G_2^+)$, it would have to be that both G_1^+ and G_2^+ were self-attacking; which cannot be, since they are grounded. If there is G' such that $G' \leadsto G^+$ but $G^+ \not\leadsto G'$, then neither G_1^+ nor G_2^+ would defend itself from attacks. So, by Theorem 4.19(i) G^+ is admissible.

Now, by Definition 4.17, G^+ is grounded iff for all maximally rule-consistent R^+, G^+ is the least fixed point of f_{R^+}, i.e., for all such R^+, $f_{\mathsf{R}^+}^{\omega}(\Diamond) = G^+$. For all such R^+, this will be true—given that $rules(G^+) \subseteq \mathsf{R}^+$—iff $v(f_{\mathsf{R}^+}^{\omega}(\Diamond)) = v(G^+)$. We show each set is included in the other.

First, $v(f_{\mathsf{R}^+}^{\omega}(\Diamond)) \subseteq v(G^+)$. For contradiction assume this is false, so that $v(f_{\mathsf{R}^+}^{\omega}(\Diamond)) \setminus v(G^+)$ is non-empty. Let k be such that $v(f_{\mathsf{R}^+}^{k}(\Diamond)) \subseteq v(G^+)$ but $v(f_{\mathsf{R}^+}^{k+1}(\Diamond)) \nsubseteq v(G^+)$, and let $a \in v(f_{\mathsf{R}^+}^{k+1}(\Diamond)) \setminus v(G^+)$; a must be an assumption, for if not then a would be in $f_{\mathsf{R}^+}^{k}(\Diamond)$. Since $a \in v(f_{\mathsf{R}^+}^{k+1}(\Diamond))$ then by the definition of f_{R^+}, where G' is such that $claim(G') = \bar{a}$, then $f_{\mathsf{R}^+}^{k}(\Diamond) \leadsto G'$. But since $f_{\mathsf{R}^+}^{k}(\Diamond) \subseteq G^+$ by hypothesis, then because $v(G^+) = v(G_1^+)$, it must be that $G_1^+ \leadsto G'$. Then since G_1^+ is grounded, $a \in v(G_1^+)$, so that $a \in v(G^+)$. Contradiction. Therefore, $v(f_{\mathsf{R}^+}^{\omega}(\Diamond)) \subseteq v(G^+)$.

Secondly, we show $v(G^+) \subseteq v(f_{\mathsf{R}^+}^{\omega}(\Diamond))$. Assume for contradiction that this is false, and thus that $v(G^+) \setminus v(f_{\mathsf{R}^+}^{\omega}(\Diamond))$ is non-empty; note further that there can be no assumption $a \in v(G^+) \setminus v(f_{\mathsf{R}^+}^{\omega}(\Diamond))$ such that $f_{\mathsf{R}^+}^{\omega}(\Diamond)$ attacks all G' with $claim(G') = \bar{a}$ (for if there were such an a, then evidently a must be in $f_{\mathsf{R}^+}^{\omega}(\Diamond)$). Now, since G^+ is admissible there must be a cycle of argument graphs $P_1, O_1, P_2, O_2, \ldots, O_m, P_1$, $m \geqslant 1$, such that $P_1 \leftsquigarrow O_1 \leftsquigarrow P_2 \leftsquigarrow O_2 \leftsquigarrow \cdots \leftsquigarrow O_m \leftsquigarrow P_1$, and for all i such that $1 \leqslant i \leqslant m$, $P_i \subseteq G^+$, $claim(P_i) \in (v(G^+) \setminus v(f_{\mathsf{R}^+}^{\omega}(\Diamond)))$, and $support(P_i) \nsubseteq f_{\mathsf{R}^+}^{\omega}(\Diamond))$. Further, for no $G' \subseteq f_{\mathsf{R}^+}^{\omega}(\Diamond))$ is it the case that for some i with $1 \leqslant i \leqslant m$, then $G' \leadsto O_i$. (These conditions are required because $v(G^+) \subseteq v(f_{\mathsf{R}^+}^{\omega}(\Diamond)))$.) Yet this cycle means that there would have to be a cycle in $\mathcal{G}_n$; since $\mathcal{G}_n$ must be acyclic, then we have a contradiction. Thus $v(G^+) \subseteq v(f_{\mathsf{R}^+}^{\omega}(\Diamond))$.

Therefore for any maximally rule-consistent R^+, G^+ is the least fixed point of f_{R^+}—i.e., G^+ is grounded.

Evidently $(G_1 \cup G_2) \subseteq G^+$ by definition: this concludes the proof. □

This concludes the second stage of preliminaries.

We now, thirdly, prove the theorem. Thus let $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$ be our GRN-graph-DD for s_0 with resulting $\mathcal{P}$ and let $\mathcal{T}$ be a corresponding argument graph dispute tree (according to Definition A.10). By Lemma A.11, $\mathcal{T}$ is finite. It is easy to see that the height of a finite argument graph dispute tree $\mathcal{T}$ is $2k$, for $k \geqslant 0$. Let $G_1, \ldots, G_m$ be the argument graphs labelling the PROPONENT nodes of $\mathcal{T}$ ($m \geqslant 1$). Evidently, $G_1 \cup \cdots \cup G_m = \mathcal{P}$, and thus by Lemma A.7, $G_1 \cup \cdots \cup G_m$ is admissible. We can show the theorem by proving, by induction on k, that there is a grounded argument graph G^* such that $G_1 \cup \cdots \cup G_m \subseteq G^*$.

We prove the result.

- **Base case**. $k = 0$ corresponds to a dispute tree with a single node labelled by an argument graph G that has no attackers. Let G' be an arbitrary grounded argument graph. Now, $\Diamond$ is an admissible argument graph (since it has no attackers), and evidently $\Diamond \subseteq G'$. Clearly, $G \nsubseteq \Diamond$ (since at least $s_0 \in v(G)$) and $\Diamond \parallel G$ (this is just G). Since G is clearly acceptable w.r.t. $\Diamond$, Lemmas A.13 and A.14 apply, giving that $\Diamond \cup G = G$ is admissible and $G \subseteq G^*$ for some grounded G^*. Thus, since here $G = \mathcal{P}$, we have shown the base case.
- **Induction step**. First assume the result holds for all finite argument graph dispute trees with height less than $2k$, and let $\mathcal{T}$ be a finite argument graph dispute tree with height $2k$, whose root is labelled by some G_r. Let $n_1, \ldots, n_j$ be the children of children of the root of $\mathcal{T}$. For each such child n_i ($1 \leqslant i \leqslant j$), let $\mathcal{T}_i$ be the finite argument graph dispute tree rooted at n_i (each $\mathcal{T}_i$ has height less than $2k$). For each such $\mathcal{T}_i$, let G_i be the union of the argument graphs labelling the PROPONENT nodes of $\mathcal{T}_i$. Then by the induction hypothesis there is, for each i, some grounded G_i^+ such that $G_i \subseteq G_i^+$. Then by Lemma A.15, there is some grounded G^+ such that $G_1 \cup \cdots \cup G_m \subseteq G^+$. Then by Lemmas A.13 and A.14, since G_r is acceptable w.r.t. $G_1 \cup \cdots \cup G_m$, it must be that $G_1 \cup \cdots \cup G_m \cup G_r$ is admissible and $G_1 \cup \cdots \cup G_m \cup G_r \subseteq G^*$ for some grounded G^*.

We conclude by induction that the result holds for all k, and this proves the theorem. □

A.9. Theorem 5.11

Let $\mathcal{L}$ be finite. If G is an admissible argument graph such that $s_0 \in v(G)$, then there is an ADM-graph-DD for s_0 with resulting argument graph some $\mathcal{P}$ such that $\mathcal{P} \subseteq G$.

Proof. We proceed as follows. We first show (Lemma A.16) that, since $\mathcal{L}$ is finite, there is no infinite (ω-length) ADM-graph-DS for s_0. We next prove that any finite ADM-graph-DS for s_0, ending with $(\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$ such that $\mathcal{P}_n$ 'matches' G (a notion we give in Definition A.17) and which satisfies three additional properties, can always be extended to a longer ADM-graph-DS which satisfies the same properties (Lemma A.19). To prove this, we use an intermediate result (Lemma A.18). Together with the result on the impossibility of infinite ADM-graph-DSs, this is proved to show the theorem.

Lemma A.16. *If $\mathcal{L}$ be finite, then there is no infinite (ω-length) X-graph-DS ($X \in \{$ADM, GRN$\}$) for $s_0 \in \mathcal{L}$.*

Proof. Suppose that there is an infinite X-graph-DS for s_0; call this S_ω. There must either be an infinite subsequence of members of S_ω determined by a PROPONENT step in Definition 5.5, or an infinite subsequence of members of S_ω determined by an OPPONENT step in Definition 5.5. Suppose the former. Note that for each PROPONENT step, some $s \notin m(\mathcal{P}_i)$ is added to $m(\mathcal{P}_i)$, and further that $m(\mathcal{P}_i)$ is non-decreasing in S_ω. Thus if there is some infinite subsequence of PROPONENT steps, $|\bigcup_{i<\omega}(m(\mathcal{P}_i)| = \aleph_0$, contradicting the finiteness of $\mathcal{L}$.

Suppose now that there is an infinite subsequence of S_ω of OPPONENT steps. Since there is no infinite subsequence of PROPONENT steps, then there must be some $j < \omega$ such that for all $i > j$, the only steps are OPPONENT. Thus consider the subsequence S_o of S_ω starting from j. Cases 2(i)(b) and 2(i)(c) mark members of $\boldsymbol{O}_i$, and case 2(i)(a) marks a member of $u(G^*)$, for some $G^* \in \boldsymbol{O}_i$; thus if there were only a finite number of steps in S_o of case 2(ii), then eventually no other case would be applicable—either because there would be no member of $u(\boldsymbol{O}_i)$, or because all members $G^* \in u(\boldsymbol{O}_i)$ would be such that $u(G^*) = \emptyset$. Therefore there are an infinite number of steps of case 2(ii) in S_o. Let us say that some $G' \in \boldsymbol{O}_{i+1}$ is a *child* of $G^* \in \boldsymbol{O}_i$, if $G^* \in u(\boldsymbol{O}_i)$ was selected at step i, and G' was added to form $u(\boldsymbol{O}_{i+1})$ by case 2(ii) (i.e., G' expands G^* at $s \in v(G_i)$ using some rule $s \leftarrow R \in \mathcal{R}$). There must then be an ω-length sequence of argument graphs $G_1, \ldots, G_n, \ldots$ such that $G_1 \in u(\boldsymbol{O}_j)$, and G_{k+1} is a child of G_k for $k = 1, \ldots, n, \ldots$. But it is not hard to see (as for the PROPONENT case, above), that this is impossible because of the finiteness of $\mathcal{L}$, given the acyclicity checks in case 2(ii).. Thus there is no infinite subsequence of OPPONENT steps in S_ω.

So there is no infinite X-graph-DS. □

Definition A.17. Let $\mathcal{P}$ be a potential argument graph, and G an argument graph. Then $\mathcal{P}$ is said to *match* G if the graph $G_\mathcal{P}$ obtained by ignoring the marking apparatus of $\mathcal{P}$ is such that $G_\mathcal{P} \subseteq G$. ⌟

Note that $G_\mathcal{P}$ is only an argument graph when $\mathcal{P}$ is actual; and that, recalling Definition 5.1, any actual argument graph matches its corresponding argument graph.

Lemma A.18. *Let $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$ be an X-graph-DS for $s_0 \in \mathcal{L}$ ($X \in \{$ADM, GRN$\}$) which is not an X-graph-DD, and let G be an admissible argument graph such that $s_0 \in v(G)$. Further suppose that, for all i such that $0 \leqslant i < n$:*

a. *if the $(i+1)$-th tuple is obtained using case 1(ii) of Definition 5.5 with $s \in (\mathcal{L} \setminus \mathcal{A})$ selected such that $s \in v(G)$, then the rule $s \leftarrow \{s' \in v(G) \mid (s, s') \in e(G)\}$ is chosen;*
b. *if the $(i+1)$-th tuple is obtained using case 2(i) of Definition 5.5 with and $G' \in u(\boldsymbol{O}_i)$ and $s \in (G' \cap \mathcal{A})$ selected, then s is ignored (case 2(i)(a) of Definition 5.5) iff $\bar{s} \notin v(G)$.*

Then for all $0 \leqslant i \leqslant n$:

i. *Let i be a PROPONENT step, where $s \notin \mathcal{A}$ is selected; then $s \in v(G)$.*
ii. *Let i be a PROPONENT step, where $a \in \mathcal{A}$ is selected; then $a \in support(G)$.*
iii. *$D_i \subseteq support(G)$.*
iv. *For each $c \in C_i$, there is $s \in v(G)$ such that $\bar{c} = s$.*
v. *Suppose i is an OPPONENT step where $G' \in u(\boldsymbol{O}_i)$ is selected, such that $u(G') \subseteq \mathcal{A}$. Then G' is a focused potential argument graph such that $claim(G') = \bar{a}$ for some $a \in support(G)$.*

Proof. First note that we are entitled to suppose (a) and (b). In particular, in the case of (a), since G is an admissible argument graph, then, where $s \in v(G)$, there must be a (unique) rule $s \leftarrow R$ in $\mathcal{R}$ such that $R = \{s' \in v(G) \mid (s, s') \in e(G)\}$. (b) makes no existential commitments and is therefore trivially permitted.

We take the proof of parts (i)–(v) in turn; that of (i)–(iii) is by induction on n.

i. The base case ($n = 0$) amounts to $s = s_0$, and is immediate. Assume the result holds for $n = k$. We will show it holds for $n = k + 1$. Thus let $s \notin \mathcal{A}$ be selected at the transition from step k to $k + 1$. Evidently s is in $v(\mathcal{P}_k)$ either by some previous case 1(ii) or 2(i)(c). If the case was 1(ii), then by condition (a) and the induction hypothesis, it must be that $s \in v(G)$. If the case was 2(i)(c), then s is in $v(\mathcal{P}_k)$ because $s = \bar{a}$ for some $a \in \mathcal{A}$ which was not ignored in the OPPONENT step; so by condition (b) and the induction hypothesis, $s \in support(G)$, and the result follows. Thus $s \in v(G)$ either way, and so by induction the result holds for all n.
ii. Similar to (i).
iii. The base case ($n = 0$) is obvious. Suppose the result holds for $n = k$. For $n = k+1$, we must show that $D_{k+1} \subseteq support(G)$. $D_k \subseteq support(G)$ through the induction hypothesis. Now, note that assumptions are added to D_k in, possibly, cases 1(ii) and 2(i)(c). If the case is 1(ii), then condition (a) guarantees that $D_{k+1} \subseteq support(G)$. If the case is 2(i)(c), then it must be that any $\bar{s}$ added is in $support(G)$, by condition (b). Thus $D_{k+1} \subseteq support(G)$, and by induction we conclude that result holds for all n.
iv. Let $c \in C_i$. Then c must have been added by some previous step of case 2(i)(c). Thus $\bar{c} \in v(G)$ by condition (b).
v. That G' is a focused potential argument graph is true by construction. There must be some $j < i$ such that $newgrph(claim(G'))$ was added to $u(\boldsymbol{O}_j)$ using 1(i) of Definition 5.5. But then evidently $claim(G') = \bar{a}$ for some $a \in v(\mathcal{P}_j) \cap \mathcal{A}$. Thus $a \in support(G)$ from part (ii) of the current lemma. □

Lemma A.19. *Let $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$ be an ADM-graph-DS for $s_0 \in \mathcal{L}$, which is not an ADM-graph-DD, such that for all i such that $0 \leqslant i < n$:*

a. *if the $(i+1)$-th tuple is obtained using case 1(ii) of Definition 5.5 with $s \in (\mathcal{L} \setminus \mathcal{A})$ selected such that $s \in v(G)$, then the rule $s \leftarrow \{s' \in v(G) \mid (s, s') \in e(G)\}$ is chosen;*
b. *if the $(i+1)$-th tuple is obtained using case 2(i) of Definition 5.5 with and $G' \in u(\boldsymbol{O}_i)$ and $s \in (G' \cap \mathcal{A})$ selected, then s is ignored (case 2(i)(a) of Definition 5.5) iff $\bar{s} \notin v(G)$;*
c. *$\mathcal{P}_i$ matches G (and so does $\mathcal{P}_n$).*

Then $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n)$ can be extended to an ADM-graph-DS for s_0

$$(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_n, \boldsymbol{O}_n, \mathcal{G}_n, D_n, C_n), (\mathcal{P}_{n+1}, \boldsymbol{O}_{n+1}, \mathcal{G}_{n+1}, D_{n+1}, C_{n+1})$$

which also satisfies conditions (a)–(c).

Proof. Note that conditions (a) and (b) are the same as in Lemma A.18; thus that lemma applies.

We show that one of the cases of Definition 5.5 must apply in such a way to preserve conditions (a)–(c), and thus that the given ADM-graph-DS can be extended. Note first that it cannot be that both $u(\mathcal{P}_n)$ and $u(\boldsymbol{O}_n)$ are empty, as otherwise the given ADM-graph-DS would be an ADM-graph-DD.

- If $u(\mathcal{P}_n) \neq \emptyset$ then either there exists $s \in u(\mathcal{P}_n) \cap \mathcal{A}$ or there exists $s \in u(\mathcal{P}_n) \setminus \mathcal{A}$, one of which is selected, and thus one of case 1(i) or case 1(ii) is applicable.
  Case 1(i) applies. Conditions (a) and (b) are trivially preserved. If condition (c) is violated, it is because $\mathcal{P}_{n+1}$ does not match G. Yet this is impossible, since $\mathcal{P}_{n+1}$ is the same as $\mathcal{P}_n$ except for the marked sentences, and $\mathcal{P}_n$ matches G.
  Case 1(ii) applies. Condition (b) is trivially preserved. To prove that conditions (a) or (c) are also preserved, we must show that there is some rule of the form $s \leftarrow \{s' \in v(G) \mid (s, s') \in e(G)\}$ in $\mathcal{R}$ such that $\{s' \in v(G) \mid (s, s') \in e(G)\} \cap C_n = \emptyset$, and $\mathcal{P}_{n+1} = updtgrph(\mathcal{P}_n, s \leftarrow \{s' \in v(G) \mid (s, s') \in e(G)\}, \emptyset)$ is acyclic and matches G.
  Now, by Lemma A.18(i), we know that $s \in v(G)$. Thus there is a rule $s \leftarrow \{s' \in v(G) \mid (s, s') \in e(G)\}$ in $\mathcal{R}$; let $R = \{s' \in v(G) \mid (s, s') \in e(G)\}$. If there is $a \in R \cap C_n$, then clearly $a \in v(G) \cap C_n$, so that by Lemma A.18(iv), $G \rightsquigarrow G$ (since $\mathcal{P}_n$ matches G, by condition (c)). Then G cannot be admissible. Contradiction. Thus $R \cap C_n = \emptyset$. Evidently, since $\mathcal{P}_n$ matches G and $updtgrph(\mathcal{P}_n, s \leftarrow R)$ only adds edges and nodes to G, $\mathcal{P}_{n+1}$ is acyclic by the acyclicity of G, and $\mathcal{P}_{n+1}$ matches G. Thus in this case too (a)–(c) are preserved.
- If $u(\mathcal{O}_n) \neq \emptyset$ then some $G' \in u(\boldsymbol{O}_n)$ can be selected. If $u(G') = \emptyset$, then by Lemma A.18(v) G' is an actual argument graph whose claim is $\bar{a}$, for some $a \in support(G)$. By condition (b), there is no $a' \in v(G')$ such that $\bar{a}' \in v(G)$; for if there were such an a', then since a' must have been selected at some previous step (since $u(G') = \emptyset$), a' would not have been ignored (because of (b), above). Therefore G cannot be admissible (since it does not counter-attack G'). Contradiction. Thus $u(G') \neq \emptyset$, $s \in u(G')$ can be selected and case 2 is applicable.
  Case 2 applies. Condition (a) is trivially preserved. We must check that conditions (b) and (c) can also be preserved.
  First, condition (b). Suppose that $s \in \mathcal{A}$. Now, either $\bar{s} \notin v(G)$ or $\bar{s} \in v(G)$. If $\bar{s} \notin v(G)$, then it is clearly safe to ignore s (since there are no additional conditions on when we may apply 2(i)(a)). If $\bar{s} \in v(G)$, then we must check that $s \notin D_n$—for if $s \in D_n$, cases 2(i)(b) or 2(i)(c) could not be applied, and the ADM-graph-DS could not be extended. But if $s \in D_n$, then by Lemma A.18(iii), $s \in support(G)$. Since $\bar{s} \in v(G)$, this means that $G \rightsquigarrow G$, and so G could not be admissible. Contradiction. So if $s \in \mathcal{A}$ and $\bar{s} \in v(G)$, then $s \notin D_i$. Thus condition (b) is satisfied.

Secondly, for condition (c), note that the difference between $\mathcal{P}_n$ and $\mathcal{P}_{n+1}$ would be because of case 2(i)(c), where $\bar{s}$ is ensured to be a node of $\mathcal{P}_{n+1}$. But since we have just shown that (b) is satisfied, then it must be that $\bar{s} \in v(G)$. Thus $\mathcal{P}_{n+1}$ must match G.
Thus for case 2, (a)–(c) are preserved. □

This concludes the preliminaries. Now note that an ADM-graph-DS of length 0, containing just a single tuple, trivially satisfies conditions (a)–(c) in Lemma A.19. Lemma A.19 states that a finite, n-length ADM-graph-DS satisfying (a)–(c) which is not an ADM-graph-DD can always be extended to an $(n+1)$-length ADM-graph-DS satisfying (a)–(c). But Lemma A.16 requires that there are no infinite-length X-graph-DSs. So extending the 0-length ADM-graph-DS in accordance with (a)–(c) must eventually result in an ADM-graph-DD. □

A.10. Theorem 5.12

Let $\mathcal{L}$ be finite. If G is a grounded argument graph such that $s_0 \in v(G)$, then there is a GRN-graph-DD for s_0 with resulting argument graph some $\mathcal{P}$ such that $\mathcal{P} \subseteq G$.

Proof. Our general strategy here is close to that in the proof of Theorem 5.11. We already know that there is no ω-length GRN-graph-DS for s_0; we will prove that any finite GRN-graph-DS for s_0 matching G which is not a GRN-graph-DD can be extended. Elements of the demonstration that the extension is possible are similar to those for ADM-graph-DSs; the added complication in the case of a GRN-graph-DS is that extension might be prevented by the failure of the acyclicity check on $\mathcal{G}_i$. (This was not an issue for ADM-graph-DSs, since the $\mathcal{G}_i$ in that case is always the empty argument graph.) We will prove that, where G is grounded, a GRN-graph-DS can be defined in such a way that the $\mathcal{G}_i$ component is acyclic. This proof borrows the structure of a related result for abstract argumentation from Thang et al. [26].

The first stage of the proof shows that a 'graph induced' (see [26]) by a 'dispute tree' (see [12]) must be acyclic. We give these background definitions next.

Definition A.20. (See Definition 5.1, [12].) A *dispute tree $\mathcal{T}$ for an argument* a_0 *w.r.t.* $(\mathit{Args}, \rightsquigarrow)$ is a rooted tree such that:

i. Every node of $\mathcal{T}$ is labelled by an argument in *Args* and is either a PROPONENT or OPPONENT node (but not both). The status of a child node is different from the status of its parent.
ii. The root is a PROPONENT node labelled by a_0.
iii. For every PROPONENT node n labelled by a, and every argument b such that $\mathsf{b} \rightsquigarrow \mathsf{a}$, there is a child of n labelled by b.
iv. For every OPPONENT node n labelled by b, there exists precisely one child of n labelled by some a such that $\mathsf{a} \rightsquigarrow \mathsf{b}$.
v. There are no other nodes except those given by (i)–(iv).

Let $\mathcal{T}$ be a dispute tree w.r.t. $(\mathit{Args}, \rightsquigarrow)$. $\mathcal{T}$ is said to be *admissible* if no $\mathsf{a} \in \mathit{Args}$ labels both a PROPONENT and OPPONENT node in $\mathcal{T}$. ⌟

For a tree $\mathcal{T}$, we let *nodes*$(\mathcal{T})$ denote the set of nodes of $\mathcal{T}$ and *edges*$(\mathcal{T})$ denote the edges. Where $\mathcal{T}$ is a dispute tree and $n \in \mathit{nodes}(\mathcal{T})$ *label*(n) is the argument which labels n. (*labels*(N) lifts this to sets of nodes N.)

It is proved in [12] that if a tree $\mathcal{T}$ is admissible, then the set A of all arguments labelling the tree's PROPONENT nodes is an admissible extension of $(\mathit{Args}, \rightsquigarrow)$.

Definition A.21. (See Definition 7, [26].) Given an admissible dispute tree $\mathcal{T}$, the *graph induced by* $\mathcal{T}$, $\mathcal{I}_{\mathcal{T}}$, has the set of nodes *labels*(*nodes*$(\mathcal{T})$), and the set of edges

$$\{(\mathsf{b}, \mathsf{a}) \mid \exists (n, n') \in \mathit{edges}(\mathcal{T}) \text{ such that } \mathit{label}(n) = \mathsf{a} \wedge \mathit{label}(n') = \mathsf{b}\}$$ ⌟

Acyclicity of $\mathcal{I}_{\mathcal{T}}$ corresponds, for finite $(\mathit{Args}, \rightsquigarrow)$ to presence in the grounded extension—as given by the following lemma.

Lemma A.22. *Given a finite* $(\mathit{Args}, \rightsquigarrow)$ *with grounded extension* A_G, $\mathsf{a}_0 \in \mathsf{A}_G$ *iff there exists a dispute tree* $\mathcal{T}$ *for* a_0 *such that* $\mathcal{I}_{\mathcal{T}}$ *is acyclic.*

Proof. This is a direct consequence of Lemmas 3 and 4 from [26]. □

Now, in general, the abstract argumentation framework $(\mathit{Args}, \rightsquigarrow)$ corresponding to our ABA framework (see [14] for a definition of this correspondence) may be infinite. (This is so even where the ABA framework is restricted to be finite in all its components—for the possibility of non-rule-minimal arguments introduces the possibility of an infinite *Args*, as in Example 6, for instance.) We accordingly define a 'restricted' abstract argumentation framework $(\mathit{Args}_G, \rightsquigarrow_G)$, making use of the given grounded argument graph G. Thus, let $(\mathit{Args}_G, \rightsquigarrow_G) \subseteq (\mathit{Args}, \rightsquigarrow)$ be such that

- $Args_G$ is the set of rule-minimal arguments a in $Args$ such that either (i) a is represented in G; or (ii) there is some $b \in support(G)$ with $claim(\mathsf{a}) = \bar{b}$.
- $\rightsquigarrow_G$ is the restriction of $\rightsquigarrow$ to $Args_G$.

Then it is clear that, since $\mathcal{L}$ is finite, so is $(Args_G, \rightsquigarrow_G)$. For, (i) the set of rule-minimal arguments represented by G is plainly finite (for G itself is finite if $\mathcal{L}$ is); and (ii) for a finite set of rule-minimal arguments, there can only be finitely many rule-minimal arguments with a given claim. Let A_G denote the set of arguments represented by G (so that $\mathsf{A}_G \subseteq Args_G$). We will prove that A_G is the grounded extension of $(Args_G, \rightsquigarrow_G)$, and given Lemma A.22 this will let us conclude that there is a finite dispute tree for an argument whose claim is s_0.

Thus, it is clear that, for each $s \in v(G)$, there is precisely one $\mathsf{a} \in \mathsf{A}_G$ such that $claim(\mathsf{a}) = s$. (Existence of at least one such argument is assured by Theorem 4.3; uniqueness is a consequence of Theorem 4.12.) Each such argument is rule-minimal, by Theorems 4.4 and 4.9. Further, A_G is not bloated, by Theorem 4.12. Let a_0 be the argument represented by G whose claim is s_0. Clearly $\mathsf{a}_0 \in \mathsf{A}_G$ and is unique and rule-minimal. The following lemma shows that the characteristic function for $(Args_G, \rightsquigarrow_G)$ and any characteristic function f_{R} such that $rules(G) \subseteq \mathsf{R}$ match in an important sense.

Lemma A.23. *Let f_{R} be any argument graph characteristic function such that $rules(G) \subseteq \mathsf{R}$, and let f be the characteristic function of $(Args_G, \rightsquigarrow_G)$. Then for all i with $0 \leqslant i$, $f^i(\emptyset)$ is the set of arguments represented in $f^i_{\mathsf{R}}(\lozenge)$.*

Proof. The proof proceeds by induction on i.

Base case. The base case is trivial: the set of arguments represented by $\lozenge$ is plainly $\emptyset$.

Inductive step. Assume the result holds for $i = j$. Evidently $f^j(\emptyset) \subseteq f^{j+1}(\emptyset)$ and $f^j_{\mathsf{R}}(\lozenge) \subseteq f^{j+1}_{\mathsf{R}}(\lozenge)$. We must show that (I) if $\mathsf{a} \in (f^{j+1}(\emptyset) \setminus f^j(\emptyset))$ then a is represented in $f^{j+1}_{\mathsf{R}}(\lozenge)$; and that (II) if a is represented in $f^{j+1}_{\mathsf{R}}(\lozenge)$ but not in $f^j_{\mathsf{R}}(\lozenge)$, then $\mathsf{a} \in f^{j+1}(\emptyset)$.

I. First assume $\mathsf{a} \in (f^{j+1}(\emptyset) \setminus f^j(\emptyset))$. Then whenever $\mathsf{b} \in Args_G$ is such that $\mathsf{b} \rightsquigarrow \mathsf{a}$, $f^j(\emptyset) \rightsquigarrow \mathsf{b}$. Let G^* be a graphical conversion of b (since b is rule minimal, G^* is just a focused argument graph where b is the largest argument represented in G^*). Let G_{a} be a graphical conversion of a (again, since a is rule-minimal, G_{a} is an argument graph which precisely matches a).
   Now, either $\mathsf{a} \in \mathsf{A}_G$ or not. We will show that if $\mathsf{a} \notin \mathsf{A}_G$ then a contradiction follows. Thus assume $\mathsf{a} \notin \mathsf{A}_G$; let $m \leqslant j$ be such that there is some $\mathsf{a}^* \notin \mathsf{A}_G$ with $\mathsf{a}^* \in (f^{m+1}(\emptyset) \setminus f^m(\emptyset))$, and there is no smaller $m' < m$ such that this is true. Pick some $\mathsf{a}^* \in (f^{m+1}(\emptyset) \setminus f^m(\emptyset))$. We have that $f^m(\emptyset)$ defends a^*, so that if $\mathsf{b} \rightsquigarrow \mathsf{a}^*$, then $f^m(\emptyset) \rightsquigarrow \mathsf{b}$. But since $\mathsf{a}^* \notin \mathsf{A}_G$, then where G_{a^*} is the argument graph corresponding to a^*, then $G_{\mathsf{a}^*} \rightsquigarrow G$, so that $G \rightsquigarrow G_{\mathsf{a}^*}$, and thus there is some argument represented in G, b, such that $\mathsf{b} \rightsquigarrow \mathsf{a}^*$. Since $\mathsf{b} \in Args_G$, then $f^m(\emptyset) \rightsquigarrow \mathsf{b}$, which means there is some $\mathsf{c} \in f^m(\emptyset)$ such that $\mathsf{c} \rightsquigarrow \mathsf{b}$; but by the inductive hypothesis, c is represented in G. Thus $G \rightsquigarrow G$, contradicting its admissibility. Thus supposing that $\mathsf{a} \notin \mathsf{A}_G$ leads to a contradiction.
   It must therefore be the case that $\mathsf{a} \in \mathsf{A}_G$. Then let G^+ be the smallest argument graph containing $f^j_{\mathsf{R}}(\lozenge)$ and the focused argument graph corresponding to a as subgraphs. (That there is such a smallest argument graph can easily be shown.) Suppose $G^* \rightsquigarrow G^+$, where without loss of generality G^* is a focused argument graph; then the argument maximally represented by G^* being b, we have that $\mathsf{b} \rightsquigarrow \mathsf{a}$ or $\mathsf{b} \rightsquigarrow f^j(\emptyset)$. In each case $f^j(\emptyset)$ attacks b, so that $f^j_{\mathsf{R}}(\lozenge) \rightsquigarrow G^*$ by the inductive hypothesis. So $G^+ \subseteq f^{j+1}_{\mathsf{R}}(\lozenge)$, and thus a is represented in $f^{j+1}_{\mathsf{R}}(\lozenge)$.

II. Assume a is represented in $f^{j+1}_{\mathsf{R}}(\lozenge)$ but not in $f^j_{\mathsf{R}}(\lozenge)$. Then wherever $\mathsf{b} \rightsquigarrow \mathsf{a}$, for some rule-minimal b, then if G_b is the focused argument graph corresponding to b, $f^j_{\mathsf{R}}(\lozenge) \rightsquigarrow G_b$. Thus by the inductive hypothesis, $f^j(\emptyset) \rightsquigarrow \mathsf{b}$, so that $\mathsf{a} \in f^{j+1}(\emptyset)$ as desired.

By induction, we conclude our result. □

This lets us prove what we desired, that A_G is the grounded extension of $(Args_G, \rightsquigarrow_G)$.

Lemma A.24. A_G *is the grounded extension of* $(Args_G, \rightsquigarrow_G)$.

Proof. G is grounded, so that by Definition 4.17, for all f_{R} such that $rules(G) \subseteq \mathsf{R}$, G is the least fixed point of f_{R}. Thus $G = f^{\omega}_{\mathsf{R}}(\lozenge)$. Let f be the characteristic function of $(Args_G, \rightsquigarrow_G)$. Now, by Lemma A.23, for all $n \in \mathbb{N}$ we have that $f^n(\emptyset)$ is the set of arguments represented in $f^n_{\mathsf{R}}(\lozenge)$. Thus $f^{\omega}(\emptyset)$ is the set of arguments represented in $f^{\omega}_{\mathsf{R}}(\lozenge)$, i.e., the set of arguments represented in G, i.e., A_G. But $f^{\omega}(\emptyset)$ is the grounded extension of $(Args_G, \rightsquigarrow_G)$. Thus A_G is this grounded extension. □

This leads to the main result of the first part of the proof.

Algorithm A.2 PRUNESEARCH($\mathcal{T}$: tree).

```
Q.push(n_0)
S.push(n_0)
P := addArg(◇, label(n_0))
while Q ≠ () do
    X := Q.pop()
    if X ∈ nodes(T) then
        A := {label(n') | ∃n'((X, n') ∈ edges(T))}
        for all a ∈ A do
            if ∀X' ∈ S, ¬∃n ∈ X' s.t. label(n) = a then
                N := {n' | (X, n') ∈ edges(T) ∧ label(n') = a}
                Q.push(N)
                S.push(N)
            end if
        end for
    else if X ∉ nodes(T) then
        while X ≠ ∅ do
            n := pickOne(X)
            X := X \ {n}
            for all n' s.t. (n, n') ∈ edges(T) do
                if claim(label(n')) ∉ v(P) then
                    Q.push(n')
                    S.push(n')
                    P := addArg(P, label(n'))
                end if
            end for
        end while
    end if
end while
return S
```

Lemma A.25. *Let G be a grounded argument graph, and suppose $s_0 \in v(G)$. Let a_0 be the argument represented by G such that $claim(\mathsf{a}_0) = s_0$. Then there is a dispute tree $\mathcal{T}$ for a_0 whose induced graph $\mathcal{I}_\mathcal{T}$ is acyclic, and is such that any argument labelling a* PROPONENT *node is represented in G.*

Proof. Let $(Args_G, \rightsquigarrow_G)$ and A_G be defined as above. $(Args_G, \rightsquigarrow_G)$ is finite. By Lemma A.24, A_G is the grounded extension of $(Args_G, \rightsquigarrow_G)$, and clearly $\mathsf{a}_0 \in \mathsf{A}_G$. Thus, by Lemma A.22, there is a dispute tree $\mathcal{T}$ for a_0 whose induced graph $\mathcal{I}_\mathcal{T}$ is acyclic. That any argument a labelling a PROPONENT node is represented in G follows from the fact that the grounded extension of $(Args_G, \rightsquigarrow_G)$ is A_G. □

The acyclicity of $\mathcal{I}_\mathcal{T}$ will be proved to correspond to the acyclicity of $\mathcal{G}_i$ in the GRN-graph-DD we will construct.

In the remainder of the proof, we show that, given a grounded argument graph G, the dispute tree $\mathcal{T}$ shown to exist in Lemma A.25 can be used to construct a GRN-graph-DD for s_0. We proceed in several stages. We first show that the tree $\mathcal{T}$ can be searched in such a way as to define a sequence $(S_0, \dots, S_m)$ whose members are either PROPONENT nodes of $\mathcal{T}$, or sets of OPPONENT nodes of $\mathcal{T}$ labelled by arguments for the same claim (Algorithm A.2). We then prove properties of the sequence $(S_0, \dots, S_m)$, in Lemma A.26. The sequence $(S_0, \dots, S_m)$ can be used to constrain choices in the definition of a GRN-graph-DS, in a way we give in Definition A.27. In Lemma A.29, the main result of the second half of this proof, we then show that these constraints imply that a finite-length GRN-graph-DS which is not a GRN-graph-DD can always be extended in such a way that the constraints of Definition A.27 are preserved.

So, let $\mathcal{T}$ be such a tree for G and s_0, and n_0 be the root of $\mathcal{T}$. We define a modified breadth-first search for $\mathcal{T}$, as Algorithm A.2.[16] If $\mathcal{T}$ is a tree as given by Lemma A.25, then we will say that any $(S_0, \dots, S_m)$ such that $(S_0, \dots, S_m) =$ PRUNESEARCH($\mathcal{T}$) is a *pruned sequence* for $\mathcal{T}$. Where $(S_0, \dots, S_m)$ is a pruned sequence for $\mathcal{T}$, let $player : \{S_0, \dots, S_m\} \to$ {PROPONENT, OPPONENT} be such that $player(S_i)$ is PROPONENT if $S_i \in nodes(\mathcal{T})$, and $player(S_i)$ is OPPONENT otherwise. Thus, $player(S_i)$ is PROPONENT iff S_i is a PROPONENT node of $\mathcal{T}$; and $player(S_i)$ is OPPONENT iff S_i is a set of OPPONENT nodes labelled by arguments for the same claim.

Lemma A.26. *Let $(S_0, \dots, S_m)$ be a pruned sequence for $\mathcal{T}$. For all i such that $0 \leqslant i < m$:*

i. *if $player(S_i) =$ PROPONENT and $label(S_i) = \mathsf{a}$:*
   a. *if $0 < i$, then there is j such that $0 < j < i$, with $player(S_j) =$ OPPONENT, and some $n \in S_j$ such that $(n, S_i) \in edges(\mathcal{T})$;*
   b. *for any b such that $\mathsf{b} \rightsquigarrow \mathsf{a}$, there is $n' \in S_j$, for some j such that $0 < j < m$ and $player(S_j) =$ OPPONENT, such that $label(n') = \mathsf{b}$;*

[16] In the algorithm, Q is a queue (initially empty) with associated operations of *push* and *pop*; R is a sequence (initially empty), where $R.push(X)$ adds X to the end of the sequence. $addArg(G, \mathsf{a})$ takes an argument graph G and returns $G \cup G_a$, where G_a is a graphical conversion of the rule-minimal argument a, and where it is presumed that $G \parallel G_a$.

c. *there is no* $j \neq i$ *such that* $player(S_j) = \text{PROPONENT}$ *and* $S_i = S_j$*;*

ii. *if* $player(S_i) = \text{OPPONENT}$*, then for any* $n \in S_i$*:*
   a. *there is some* j *such that* $j < i$*, and* $(S_j, n) \in edges(\mathcal{T})$*;*
   b. *there is some* S_j *such that* $0 \leqslant j \leqslant m$ *and* $player(S_j) = \text{PROPONENT}$*, such that* $label(n') \rightsquigarrow label(n)$*;*
   c. S_i *is* $\{n' \in nodes(\mathcal{T}) \mid claim(label(n')) = claim(label(n))\}$*, and for all* S_j*, for* $j \neq i$ *and* $player(S_j) = \text{OPPONENT}$*, then* $S_i \cap S_j = \emptyset$*.* ⌟

Proof. We prove the result by induction on i.

Base case: $i = 0$. Since $player(S_0) = \text{PROPONENT}$, we just need to check (i)(b). Thus let $\mathsf{b} \rightsquigarrow \mathsf{a}$. When S_0 is popped from Q at line 5 of Algorithm A.2, then A (line 7) will contain all b' s.t. $\mathsf{b}' \rightsquigarrow \mathsf{a}$. Then clearly b is the label of some $n' \in S_j$ such that $0 < j$, and $player(S_j) = \text{OPPONENT}$. Evidently, since from line 3, $claim(label(n_0)) \in v(G)$, there can be no S_j with $player(S_j) = \text{PROPONENT}$, and $i \neq j$, such that $S_i = S_j$, since line 23 of the algorithm prevents it.

Induction step. Assume true for $i = j$; we will show that the result follows for $i = j + 1$. Thus suppose $j + 1 < m$, and first suppose that $player(S_{j+1}) = \text{PROPONENT}$; then clearly $S_{j+1} \in nodes(\mathcal{T})$, and the only way S_{j+1} could have been added was because there is $(n', S_{j+1}) \in edges(\mathcal{T}))$ such that $n' \in S_{j'}$ for $j' < j + 1$ and $player(S_{j'}) = \text{OPPONENT}$; thus condition (i)(a) holds. Further, let $label(S_{j+1})$ be a and $\mathsf{b} \rightsquigarrow \mathsf{a}$. Argumentation as for the base case shows that there is $n' \in S_{j'}$, for some j' such that $0 < j' < m$ and $player(S_{j'}) = \text{OPPONENT}$, such that $label(n') = \mathsf{b}$. Thus (i)(b) holds. Finally, line 23 of Algorithm A.2 evidently ensures that (i)(c) holds.

Suppose instead that $player(S_{j+1}) = \text{OPPONENT}$ and $n \in S_{j+1}$. Then S_{j+1} must have been added at line 12 of Algorithm A.2, and thus there is $j' < j + 1$ such that $player(S_{j'}) = \text{PROPONENT}$ and $(S_{j'}, n) \in edges(\mathcal{T})$, thus (ii)(a) holds. Further, line 22 guarantees that (ii)(b) is assured. Plainly, lines 9–10 of the algorithm ensure that condition (ii)(c) is satisfied.

Thus the inductive step holds, and we conclude the result by induction. □

It is an obvious corollary of Lemma A.26 that, where $\mathcal{T}^-$ is the graph such that $nodes(\mathcal{T}^-)$ are

$$\{S_i \mid 0 \leqslant i \leqslant m, player(S_i) = \text{PROPONENT}\}$$
$$\cup \ \{n \mid \exists S_i \in \{S_0, \ldots, S_m\}, player(S_i) = \text{OPPONENT}, n \in S_i\}$$

and whose edges $edges(\mathcal{T}^-)$ are

$$\{(n, n') \mid n, n' \in nodes(\mathcal{T}^-), (n, n') \in edges(\mathcal{T})\},$$

then $\mathcal{T}^-$ is a subtree of $\mathcal{T}$, and $\mathcal{T}^-$ is acyclic because $\mathcal{T}$ is acyclic.

In the following we define a certain sequence of tuples, and associated functions. One of those function names is *player*; we trust this will not give rise to any ambiguity (since its use is similar to that described above).

Definition A.27. Let $(S_0, \ldots, S_m)$ be a pruned search for $\mathcal{T}$, such that s_0 is $claim(label(S_0))$. Any sequence $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_K, \boldsymbol{O}_K, \mathcal{G}_K, D_K, C_K)$, such that there are functions $f_S : \{0, \ldots, K\} \to \{S_0, \ldots, S_m, \top_0, \ldots, \top_m, \top\}$ and $player : \{0, \ldots, K\} \to \{\text{PROPONENT}, \text{OPPONENT}\}$, and the following holds, is known as a GRN-*graph-DS for* s_0 *constrained by* $(S_0, \ldots, S_m)$.

i. $f_S(0) = S_0$, $player_0 = \text{PROPONENT}$ and:

$$\mathcal{P}_0 = newgrph(s_0)$$
$$\boldsymbol{O}_0 = \emptyset$$
$$\mathcal{G}_0 = \begin{cases} \Diamond & \text{if } X = \text{ADM} \\ (\{s_0\}, \emptyset) & \text{if } X = \text{GRN} \end{cases}$$
$$D_0 = \mathcal{A} \cap \{s_0\}$$
$$C_0 = \emptyset$$

ii. The $(i + 1)$th tuple is defined according to Definition 5.5 (for $X = \text{GRN}$) where for all i such that $0 \leqslant i < K$, then
   - if $u(\mathcal{P}_{i+1}) = \emptyset$ and $u(\boldsymbol{O}_{i+1}) = \emptyset$, then $player_{i+1} = \text{PROPONENT}$ and $f_S(i + 1) = \top$.

iii. For all i such that $0 \leqslant i < K$, if $player_i = \text{PROPONENT}$, then $f_S(i) = S_j$ for some $0 \leqslant j \leqslant m$ such that $S_j \in nodes(\mathcal{T})$ and $s \in u(\mathcal{P}_i) \cap labels(nodes(label(S_j))$ is selected such that
   - if $u(\mathcal{P}_i) \cap (\mathcal{L} \setminus \mathcal{A}) \neq \emptyset$, then (I) $s \in (\mathcal{L} \setminus \mathcal{A})$ and (II) in case 1(ii) of Definition 5.5, the rule $s \leftarrow \{s' \in v(G) \mid (s, s') \in e(G)\}$ is selected.

   Further:
   - if $u(\mathcal{P}_{i+1}) \cap labels(nodes(label(S_j))) \neq \emptyset$: $player_{i+1} = \text{PROPONENT}$ and $f_S(i + 1) = f_S(i)$;
   - if $u(\mathcal{P}_{i+1}) \cap labels(nodes(label(S_j))) = \emptyset$:
     – if $u(\mathcal{P}_{i+1}) \neq \emptyset$ then $player_{i+1} = \text{PROPONENT}$ and $f_S(i + 1) = S_{j+1}$;
     – if $u(\mathcal{P}_{i+1}) = \emptyset$

  - ∗ if there is $G \in u(\boldsymbol{O}_{i+1})$ such that there is an actual argument graph G^+ with $G \subseteq G^+$, then $player_{i+1} = \text{OPPONENT}$, $f_S(i+1) = S_{j+1}$, and there is $n \in S_{j+1}$ such that $label(n)$ is represented by G^+ for such a G^+;
  - ∗ if $u(\boldsymbol{O}_{i+1}) \neq \emptyset$ and there is no $G \in u(\boldsymbol{O}_{i+1})$ such that there is an actual argument graph G^+ with $G \subseteq G^+$, then $player_{i+1} = \text{OPPONENT}$ and $f_S(i+1) = \top_j$.

iv. For all i such that $0 \leqslant i < K$, if $player_i = \text{OPPONENT}$:

- $f_S(i) = S_j$ for some $0 < j < m$ iff there is $G \in u(\boldsymbol{O}_i)$ such that [there is an actual argument graph G' with $G \subseteq G'$ and some $n \in f_S(i)$ with $claim(n) = claim(G)$]
- if there is $G \in u(\boldsymbol{O}_i)$ such that [there is an actual argument graph G' with $G \subseteq G'$ and some $n \in f_S(i)$ with $claim(n) = claim(G)$], then such a G is selected;

$s \in u(G)$ is selected such that

- (I) if $u(G) \cap (\mathcal{L} \setminus \mathcal{A}) \neq \emptyset$, then $s \in (\mathcal{L} \setminus \mathcal{A})$, or (II) if $u(G) \cap (\mathcal{L} \setminus \mathcal{A}) = \emptyset$, then there is an argument a represented by G and some $n \in f_S(i)$ such that $\mathsf{a} = label(n)$, and $s \in (u(\boldsymbol{O}_i) \cap \mathcal{A})$ is selected such that there is $n' \in \bigcup_{0 \leqslant j \leqslant m}\{S_j \mid player(S_j) = \text{PROPONENT}\}$ such that $(n, n') \in edges(\mathcal{T})$ and $claim(label(n')) = \bar{s}$, and the case is *not* 2(i)(a) (i.e., s is not ignored).

Further:

- if $u(\boldsymbol{O}_{i+1}) = \emptyset$, $u(\mathcal{P}_{i+1}) \neq \emptyset$, and either $f_S(i) = S_j$ or $f_S(i) = \top_j$, then $player_{i+1} = \text{PROPONENT}$ and $f_S(i+1) = S_{j+1}$.
- if $u(\boldsymbol{O}_{i+1}) \neq \emptyset$:
  - – if there is $G' \in u(\boldsymbol{O}_{i+1})$ such that [there is some G^+ which is actual and $G' \subseteq G^+$]:
    - ∗ if there is $G' \in u(\boldsymbol{O}_{i+1})$ such that [$claim(G') = claim(G)$ and there is some G^+ such that G^+ is actual and $G' \subseteq G^+$], then $player_{i+1} = \text{OPPONENT}$, $f_S(i+1) = f_S(i)$, and there is some $n \in f_S(i+1)$ such that $label(n)$ is represented in G^+ for such a G^+;
    - ∗ if there is no $G' \in u(\boldsymbol{O}_{i+1})$ such that [$claim(G') = claim(G)$ and there is some G^+ such that G^+ is actual and $G' \subseteq G^+$], and where $f_S(i) = S_j$, then $player_{i+1} = \text{OPPONENT}$, $f_S(i+1) = S_{j+1}$, and there is some $n \in f_S(i+1)$ such that $label(n)$ is represented by G^+ for such a G^+;
  - – if there is no $G' \in u(\boldsymbol{O}_{i+1})$ such that [there is some G^+ which is actual and $G' \subseteq G^+$], and either $f_S(i) = \top_j$ or $f_S(i) = S_j$, then $player_{i+1} = \text{OPPONENT}$ and $f_S(i+1) = \top_j$. ⌟

That Definition A.27 is sound in that it defines a special case of a GRN-graph-DS easily follows.

Lemma A.28. *Let $(S_0, \ldots, S_m)$ be a pruned search for a grounded tree $\mathcal{T}$, and let s_0 be $claim(label(S_0))$. (i) Any GRN-graph-DS for s_0 constrained by $(S_0, \ldots, S_m)$ is also a GRN-graph-DS for s_0. (ii) Each $\mathcal{P}_i$ of the GRN-graph-DS (constrained or otherwise) matches G.*

Proof.

i. Trivially true, given Definitions A.27 and 5.5. Condition (i) of Definition A.27 specifies the same form of initial tuple, and condition (ii) of that definition insists that the $(i+1)$th tuple is defined according to Definition 5.5. Conditions (iii) and (iv) simply constrain that strategy for the various choice-points in Definition 5.5.

ii. Evidently $\mathcal{P}_0$ matches G, since $s_0 \in v(G)$; and if any $\mathcal{P}_i$ matches G, then condition (iii) of Definition A.27 shows that $\mathcal{P}_{i+1}$ must match G, too. The result follows by induction. □

Now note that condition (iii) of Definition A.27 evidently implies condition (a) of Lemma A.19. For if $s \in (\mathcal{L} \setminus \mathcal{A})$ is chosen in case 1(ii), then it must be that $s \in v(G)$ by construction of $\mathcal{T}$ given Lemma A.25. Further, condition (iv) of Definition A.27 implies condition (b) of Lemma A.19. What is required for this implication is that if the $(i+1)$th tuple is defined using case 2(i) of Definition 5.5, and $G' \in u(\boldsymbol{O}_i)$ and $s \in (G' \cap \mathcal{A})$ are selected, then s is ignored iff $\bar{s} \notin v(G)$. Yet if $s \in (G' \cap \mathcal{A})$ is selected, then according to condition (iv) of Definition A.27, s is *never* ignored. We must therefore show that in this case, $\bar{s} \in v(G)$. Let a be the rule-minimal argument represented by the argument graph matching G' such that $claim(\mathsf{a}) = claim(G')$. Then (iv) above insists that there must be $n \in S_i$ and $(n, n') \in edges(\mathcal{T})$ such that $claim(label(n')) = \bar{s}$. Yet since $player(S_i) = \text{PROPONENT}$, then given the definition of $(S_0, \ldots, S_m)$, $claim(label(n')) = claim(S_i) = \bar{s}$, so that $\bar{s}$ is the claim of an argument labelling a PROPONENT node of $\mathcal{T}$. Thus $\bar{s} \in v(G)$ by Lemma A.25.

Therefore conditions (a) and (b) of Lemma A.19 are satisfied, condition (c) is satisfied by Lemma A.28, and so Lemma A.19 applies. Given this background, we now move to show that any finite GRN-graph-DS for s_0 constrained by $(S_0, \ldots, S_m)$, which is not a GRN-graph-DS, can always be extended. (This result corresponds to Lemma A.19 in the proof of the completeness for admissible semantics, i.e., the proof of Theorem 5.11.) Given the fact that are no ω-length GRN-graph-DSs, that will give us our result.

Lemma A.29. *Let $(S_0, \ldots, S_m)$ be a pruned search for a grounded tree $\mathcal{T}$, and let s_0 be $claim(label(S_0))$. Let $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_K, \boldsymbol{O}_K, \mathcal{G}_K, D_K, C_K)$ be a GRN-graph-DS for s_0 constrained by $(S_0, \ldots, S_m)$, which is not a GRN-graph-DD. Then this can be extended to a GRN-graph-DS for s_0 constrained by $(S_0, \ldots, S_m)$ of the form:*

$$(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_K, \boldsymbol{O}_K, \mathcal{G}_K, D_K, C_K), (\mathcal{P}_{K+1}, \boldsymbol{O}_{K+1}, \mathcal{G}_{K+1}, D_{K+1}, C_{K+1}).$$

Proof. There must exist functions $player : \{0, \dots, K\} \to \{\text{PROPONENT}, \text{OPPONENT}\}$ and $f_S : \{0, \dots, K\} \to \{S_0, \dots, S_m, \top_0, \dots, \top_m, \top\}$ with the properties given in Definition A.27. We will show that the tuple $(\mathcal{P}_{K+1}, \boldsymbol{O}_{K+1}, \mathcal{G}_{K+1}, D_{K+1}, C_{K+1})$ and functions $player^+ : \{0, \dots, K, K+1\} \to \{\text{PROPONENT}, \text{OPPONENT}\}$ and $f_S^+ : \{0, \dots, K, K+1\} \to \{S_0, \dots, S_m, \top_0, \dots, \top_m, \top\}$ can be given which satisfy Definition A.27. Since our new functions will match the originals for $\{0, \dots, K\}$, we simply need to define the values $player^+_{K+1}$ and $f_S^+(K+1)$ (as well as the $(K+1)$th tuple).

We proceed by cases.

- Suppose $player_K = \text{PROPONENT}$, and let $f_S(K) = S_j$. Here, it must be that $u(\mathcal{P}_K) \neq \emptyset$. $\mathcal{P}_K$ matches G.
  We first consider how to define $player^+_{K+1}$ and $f_S^+(K+1)$.
  - If $u(\mathcal{P}_{K+1}) \neq \emptyset$, then set $player^+_{K+1} = \text{PROPONENT}$. If $u(\mathcal{P}_{K+1}) \cap labels(nodes(label(f_S(K)))) = \emptyset$ then set $f^+_{K+1} = S_{j+1}$; or if, alternatively, we have $u(\mathcal{P}_{K+1}) \cap labels(nodes(label(f_S(K)))) \neq \emptyset$ then set $f^+_{K+1} = S_j$.
  - If $u(\mathcal{P}_{K+1}) = \emptyset$ and $u(\boldsymbol{O}_{K+1}) = \emptyset$, then set $player^+_{K+1} = \text{PROPONENT}$ and $f_S^+(K+1) = \top$.
  - If $u(\mathcal{P}_{K+1}) = \emptyset$ and $u(\boldsymbol{O}_{K+1}) \neq \emptyset$, then set $player^+_{K+1} = \text{OPPONENT}$, and if there is $G' \in u(\boldsymbol{O}_{K+1})$ such that there is an actual argument graph G^+ with $G' \subseteq G^+$, then set $f_S^+(K+1) = S_{j+1}$; or if there is no such G', set $f_S^+(K+1) = \top_j$.

  Now, we must show that the sequence *can* be extended. This will only be prevented when $\mathcal{G}_{i+1}$ has a cycle, so that we must show $acyclic(\mathcal{G}_{i+1})$. There are now two cases. Either (i) $u(\mathcal{P}_K) \cap (\mathcal{L} \setminus \mathcal{A}) \neq \emptyset$, or (ii) $u(\mathcal{P}_K) \cap (\mathcal{L} \setminus \mathcal{A}) = \emptyset$. If (i), then we select some $s \in u(\mathcal{P}_K) \cap (\mathcal{L} \setminus \mathcal{A})$, and since $\mathcal{P}_K$ matches G, we choose $s \leftarrow R$ such that $R = \{s' \mid (s, s') \in e(G)\}$ (such an $s \leftarrow R$ plainly exists). If (ii), we already know that the $s \in u(\mathcal{P}_K)$ selected must be in $v(G)$. Thus in either case, $\mathcal{P}_{K+1}$ matches G, and so the only way that the sequence could fail to be extended to the $(K+1)$th tuple would be if it were not the case that $acyclic(\mathcal{G}_{K+1})$.

  Thus suppose for contradiction that there is a cycle in $\mathcal{G}_{i+1}$. Since $\mathcal{P}_{K+1}$ is acyclic (since it matches G, and G must be acyclic), any cycle in $\mathcal{G}_{K+1}$ must pass through some $s_1 \notin v(\mathcal{P}_{K+1})$; thus let $s_1, \dots, s_r, s_1$ be a cycle in $\mathcal{G}_{K+1}$, for $s_1 \notin v(\mathcal{P}_{K+1})$. From examination of the construction of $\mathcal{G}_{K+1}$ in Definition 5.5, we can structure the cycle in the form

  $$[o_1][p_1^1, \dots, p_1^{k(1)}][o_2][p_2^1, \dots, p_2^{k(2)}] \cdots [p_l^1, \dots, p_l^{k(l)}][o_1]$$

  such that for all i with $1 \leqslant i \leqslant l$, $k(i) \geqslant 1$, and

  $$(o_1, p_1^1, \dots, p_1^{k(1)}, o_2, p_2^1, \dots, p_2^{k(2)}, \dots, p_l^1, \dots, p_l^{k(l)}, o_1) = (s_1, \dots, s_r, s_1)$$

  and where, for all j such that $1 \leqslant j \leqslant l$, and given that the selection function always chooses non-assumptions over assumptions where possible—as Definition A.27 shows—the following hold.

  a. Each $[p_j^1, \dots, p_j^{k(j)}]$ corresponds to the unique argument a_j represented by G such that $claim(\mathsf{a}_j) = p_j^1$.
     The nature of the correspondence is as follows. Evidently $p_j^1 \in v(G)$, so let a_j be the argument a_j represented by G such that $claim(\mathsf{a}_j) = p_j^1$. This is clearly precisely one such a_j, since G is an argument graph—Theorem 4.3 ensures existence of a_j and Theorem 4.12 shows uniqueness. The sequence $[p_j^1, \dots, p_j^{k(j)}]$ is a path from the claim of the argument to a member of the arguments support, such that each p_j^h for $1 < h \leqslant k(j)$ is in the body of rule whose head is p_j^{h-1}.
  b. Each o_j corresponds to an argument b_j not represented by G.
     The use of a *patient selection function* means that there can only be an edge $(o_j, p_j^1) \in \mathcal{G}_{K+1}$ if, at the point this edge was added, the $G' \in \boldsymbol{O}$ selected was such that G' had the structure of a focused argument graph (that is: with the marking apparatus removed, G' is a focused argument graph). b_j is the argument uniquely represented by this argument graph. (For the existence and uniqueness, see (a), above.)
  c. $\mathsf{a}_j \rightsquigarrow \mathsf{b}_j$, and if $j < l$, $\mathsf{b}_{j+1} \rightsquigarrow \mathsf{a}_j$, and $\mathsf{b}_l \rightsquigarrow \mathsf{a}_{n-1}$.
     The structure of attacks is clear from the nature of Definition 5.5.

  Then, it is not hard to see that (c) above means that there is a cycle of attacks of arguments

  $$\mathsf{a}_l \rightsquigarrow \mathsf{b}_l \rightsquigarrow \mathsf{a}_{l-1} \rightsquigarrow \mathsf{b}_{l-1} \rightsquigarrow \cdots \rightsquigarrow \mathsf{a}_1 \rightsquigarrow \mathsf{b}_1 \rightsquigarrow \mathsf{a}_l$$

  in $\mathcal{I}_\mathcal{T}$. But given Theorem 4.20(iii), this means G cannot be grounded. Contradiction. Therefore $\mathcal{G}_{K+1}$ is acyclic.

  It now remains to show that the extended sequence, together with $player^+$ and f_S^+ defined as above, satisfy Definition A.27. Evidently condition (i) is satisfied by hypothesis. Condition (ii) is satisfied by construction. The first bullet of condition (iii) is satisfied given our way of choosing $s \leftarrow R$ in a way consistent with the structure of G; the remaining bullets are satisfied by construction. Condition (iv) is trivially satisfied, since $player^+_{K+1} = \text{PROPONENT}$.
- Suppose instead that $player_K = \text{OPPONENT}$, so that $u(\boldsymbol{O}_K) \neq \emptyset$. As for the PROPONENT case, we are at liberty to define $player^+_{K+1}$ and $f_S^+(K+1)$ in such a way that condition (4) of Definition A.27 is satisfied. We omit the details, which can be directly copied from the definition.
  Now, we must show, as for the PROPONENT case, that the sequence can be extended so as to satisfy:
  i. $f_S(K) = S_j$ for some $0 < j < m$ iff there is $G \in u(\boldsymbol{O}_K)$ such that [there is an actual argument graph G' with $G \subseteq G'$ and some $n \in f_S(K)$ with $claim(n) = claim(G)$]

ii. if there is $G \in u(\boldsymbol{O}_K)$ such that [there is an actual argument graph G' with $G \subseteq G'$ and some $n \in f_S(K)$ with $claim(n) = claim(G)$], then such a G is selected;

$s \in u(G)$ is selected such that

iii. (a) if $u(G) \cap (\mathcal{L} \setminus \mathcal{A}) \neq \emptyset$, then $s \in (\mathcal{L} \setminus \mathcal{A})$, or (b) if $u(G) \cap (\mathcal{L} \setminus \mathcal{A}) = \emptyset$, then there is an argument a represented by G and some $n \in f_S(K)$ such that $\mathrm{a} = label(n)$, and $s \in (u(\boldsymbol{O}_K) \cap \mathcal{A})$ is selected such that there is $n' \in \bigcup_{0 \leqslant j \leqslant m}\{S_j \mid player(S_j) = \textsc{proponent}\}$ such that $(n, n') \in edges(\mathcal{T})$ and $claim(label(n')) = \bar{s}$, and the case is *not* 2(i)(a) (i.e., s is not ignored).

Conditions (ii) and (iii)(a) may be trivially satisfied (by stipulation of how to define the $(K+1)$th tuple).

First consider (i), and suppose initially that $f_S(K) = S_j$ for some $0 < j < m$. Since the sequence $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_K, \boldsymbol{O}_K, \mathcal{G}_K, D_K, C_K)$ is a GRN-graph-DS for s_0 constrained by the sequence $(S_0, \ldots, S_m)$, Definition A.27 applies. If $player_{K-1} = \textsc{proponent}$, then case (iii) of that definition requires that, since $player_K = \textsc{opponent}$ and $f_S(K) = S_j$, then there is $G \in \boldsymbol{O}_K$ such that there is an actual argument graph G' with $G \subseteq G'$, and some $n \in f_S(K)$ such that $claim(G') = claim(label(n))$, so that evidently $claim(label(n)) = claim(G)$, as required. Alternatively suppose that $player_{K-1} = \textsc{opponent}$. Then an examination of case (iv) of Definition A.27 shows that, whether $f_S(K-1) = f_S(K)$ or not, it must be that there is $G \in u(\boldsymbol{O}_K)$ such that there is an actual argument graph G' with $G \subseteq G'$, and some $n \in f_S(K)$ such that $claim(G') = claim(label(n))$.

Suppose instead that there is some $G \in u(\boldsymbol{O}_K)$ such that there is an actual argument graph G' with $G \subseteq G'$ and some $n \in f_S(K)$ with $claim(n) = claim(G)$. A similar argument to that in the previous paragraph, in the other direction, means that there must be some j with $0 < j < m$ such that $f_S(K) = S_j$.

Now consider (iii)(b). Suppose that $s \in u(G) \cap \mathcal{A}$ is selected. Since the selection function for members of $u(G)$ is patient, it must be that G matches a focused argument graph which represents an argument a, such that the claim of that focused argument graph and that of a are identical. If $|labels(nodes(\mathrm{a}))| > 1$, then there would be $G' \in u(\boldsymbol{O}_J)$ for some $J < K$ such that $G' \subseteq G$, and case (iv) of Definition A.27 requires that there is $f_S(J) = f_S(K)$, and some $n \in f_S(K)$ such that $label(n)$ is a, as required. If a consists of a single assumption, then similar argumentation based on cases (iii) or (iv) (depending on whether $player_{K-1}$ is PROPONENT or OPPONENT) shows that there is again some $n \in f_S(K)$ such that $label(n)$ is a.

Then, since there is such a $label(n)$ in $f_S(K)$, Lemma A.26 shows that there must be some S_j with $0 \leqslant j \leqslant m$, with $label(S_j) \rightsquigarrow label(n)$. Let $a \in label(n)$ be such that $claim(label(S_j)) = \bar{a}$; then evidently $a \notin D_K$, thus we are free not to ignore this case, i.e., we need not choose 2(i)(a).

Thus conditions (i) and (iii)(b) may be satisfied; the only barrier to extending the K-length sequence would then be a cycle in G_{K+1}; but the argument we made above for the case where $player_K = \textsc{proponent}$, to show that G_{K+1}, applies in just the same way if $player_K = \textsc{opponent}$.

Thus, in both cases, if $(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_K, \boldsymbol{O}_K, \mathcal{G}_K, D_K, C_K)$ is not a GRN-graph-DD, and is a GRN-graph-DS for s_0 constrained by $(S_0, \ldots, S_m)$, we can extend it to a GRN-graph-DS

$$(\mathcal{P}_0, \boldsymbol{O}_0, \mathcal{G}_0, D_0, C_0), \ldots, (\mathcal{P}_K, \boldsymbol{O}_K, \mathcal{G}_K, D_K, C_K), (\mathcal{P}_{K+1}, \boldsymbol{O}_{K+1}, \mathcal{G}_{K+1}, D_{K+1}, C_{K+1})$$

for s_0 also constrained by $(S_0, \ldots, S_m)$. □

It is now a short step to our result. Since Lemma A.16 gives us that there is no ω-length GRN-graph-DS, it must be, given Lemma A.29, that any GRN-graph-DS for s_0 constrained by $(S_0, \ldots, S_m)$ of length 0 (i.e., containing only a single tuple) can always be extended to a GRN-graph-DD. Yet plainly, a GRN-graph-DS for s_0 of length 0 exists: the single tuple of which it consists was defined as item 1 of Definition A.27. □

References

[1] L. Amgoud, The outcomes of logic-based argumentation systems under preferred semantics, in: E. Hüllermeier, S. Link, T. Fober, B. Seeger (Eds.), Proceedigs of the 6th International Conference on Scalable Uncertainty Management, SUM 2012, Springer, 2012, pp. 72–84.

[2] T.J. Bench-Capon, P.E. Dunne, Argumentation in artificial intelligence, Artif. Intell. 171 (2007) 619–641.

[3] P. Besnard, A.J. García, A. Hunter, S. Modgil, H. Prakken, G.R. Simari, F. Toni, Introduction to structured argumentation, Argum. Comput. 5 (2014) 1–4.

[4] P. Besnard, A. Hunter, Elements of Argumentation, MIT Press, 2008.

[5] P. Besnard, A. Hunter, Constructing argument graphs with deductive arguments: a tutorial, Argum. Comput. 5 (2014) 5–30.

[6] A. Bondarenko, P.M. Dung, R.A. Kowalski, F. Toni, An abstract, argumentation-theoretic approach to default reasoning, Artif. Intell. 93 (1997) 63–101.

[7] M. Caminada, Semi-stable semantics, in: P.E. Dunne, T.J. Bench-Capon (Eds.), Computational Models of Argument: Proceedings of COMMA, Liverpool, UK, September 11–12, 2006, IOS Press, 2006, pp. 121–130.

[8] R. Craven, F. Toni, A. Hadad, C. Cadar, M. Williams, Efficient support for medical argumentation, in: G. Brewka, T. Eiter, S.A. McIlraith (Eds.), Proc. 13th International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, 2012, pp. 598–602.

[9] R. Craven, F. Toni, M. Williams, Graph-based dispute derivations in assumption-based argumentation, in: E. Black, S. Modgil, N. Oren (Eds.), Theory and Applications of Formal Argumentation, Springer, 2013, pp. 46–62.

[10] Y. Dimopoulos, B. Nebel, F. Toni, On the computational complexity of assumption-based argumentation for default reasoning, Artif. Intell. 141 (2002) 57–78.

[11] P.M. Dung, On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games, Artif. Intell. 77 (1995) 321–357.

[12] P.M. Dung, R.A. Kowalski, F. Toni, Dialectic proof procedures for assumption-based, admissible argumentation, Artif. Intell. 170 (2006) 114–159.

[13] P.M. Dung, R.A. Kowalski, F. Toni, Assumption-based argumentation, in: I. Rahwan, G.R. Simari (Eds.), Argumentation in AI, Springer, 2009, pp. 25–44.
[14] P.M. Dung, P. Mancarella, F. Toni, Computing ideal sceptical argumentation, Artif. Intell. 171 (2007) 642–674.
[15] P.E. Dunne, The computational complexity of ideal semantics, Artif. Intell. 173 (2009) 1559–1591.
[16] V. Efstathiou, A. Hunter, Algorithms for effective argumentation in classical propositional logic: a connection graph approach, in: S. Hartmann, G. Kern-Isberner (Eds.), Foundations of Information and Knowledge Systems, in: Lecture Notes in Computer Science, vol. 4932, Springer, 2008, pp. 272–290.
[17] U. Egly, S.A. Gaggl, S. Woltran, Answer-set programming encodings for argumentation frameworks, Argum. Comput. 1 (2010) 147–177.
[18] X. Fan, R. Craven, R. Singer, F. Toni, M. Williams, Assumption-based argumentation for decision-making with preferences: a medical case study, in: J. Leite, T.C. Son, P. Torroni, L. van der Torre, S. Woltran (Eds.), Proceedings of the 14th International Workshop on Computational Logic in Multi-Agent Systems, CLIMA XIV, Corunna, Spain, September 16–18, 2013, Springer, 2013, pp. 374–390.
[19] X. Fan, F. Toni, A. Mocanu, M. Williams, Dialogical two-agent decision making with assumption-based argumentation, in: A. Lomuscio, P. Scerri, A. Bazzan, M. Huhns (Eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014, IFAAMAS, 2014, pp. 533–540.
[20] A.J. García, G.R. Simari, Defeasible logic programming: an argumentative approach, Theory Pract. Log. Program. 4 (2004) 95–138.
[21] R. Kowalski, A proof procedure using connection graphs, J. ACM 22 (1975) 572–595.
[22] P.A. Matt, F. Toni, T. Stournaras, D. Dimitrelos, Argumentation-based agents for eprocurement, in: M. Berger, B. Burg, S. Nishiyama (Eds.), Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, Industry and Applications Track, AAMAS 2008, 2008, pp. 71–74.
[23] S. Modgil, H. Prakken, A general account of argumentation with preferences, Artif. Intell. 195 (2013) 361–397.
[24] I. Rahwan, G.R. Simari, Argumentation in Artificial Intelligence, Springer, 2009.
[25] M. Snaith, C. Reed, TOAST: online ASPIC$^+$ implementation, in: B. Verheij, S. Szeider, S. Woltran (Eds.), Proceedings of the Fourth International Conference on Computational Models of Argument, COMMA 2012, Vienna, Austria, September 10–12, 2012, IOS Press, 2012, pp. 509–510.
[26] P.M. Thang, P.M. Dung, N.D. Hung, Towards a common framework for dialectical proof procedures in abstract argumentation, J. Log. Comput. 19 (2009) 1071–1109.
[27] F. Toni, A generalised framework for dispute derivations in assumption-based argumentation, Artif. Intell. 195 (2013) 1–43.
[28] F. Toni, A tutorial on assumption-based argumentation, Argum. Comput., Special Issue: Tutorials on Structured Argumentation 5 (2014) 89–117.
[29] G. Vreeswijk, Abstract argumentation systems, Artif. Intell. 90 (1997) 225–279.
[30] Q. Zhong, X. Fan, F. Toni, X. Luo, Explaining best decisions via argumentation, in: A. Herzig, E. Lorini (Eds.), Proceedings of the European Conference on Social Intelligence, ECSI-2014, Barcelona, Spain, November 3–5, 2014, CEUR-WS.org, 2014, pp. 224–237.