Graph-Based Dispute Derivations in Assumption-Based Argumentation

Robert Craven[1], Francesca Toni[1] and Matthew Williams[2]

[1] Department of Computing / [2] Faculty of Medicine,
Imperial College London
{robert.craven,ft}@imperial.ac.uk, mhw@doctors.org.uk

Abstract. Arguments in structured argumentation are usually defined as trees. This introduces both conceptual redundancy and inefficiency in standard methods of implementation. We introduce *rule-minimal arguments* and *argument graphs* to solve these problems, studying their use in assumption-based argumentation (ABA), a well-known form of structured argumentation. In particular, we define a new notion of *graph-based dispute derivations* for determining acceptability of claims under the grounded semantics in ABA, study formal properties and present an experimental evaluation thereof.

1 Introduction

Assumption-Based Argumentation (ABA) [1–5] is a well-known framework for structured argumentation where, in contrast to abstract argumentation [6], arguments and attacks are not primitives but are derived from the rules of a given deductive system, assumptions and contraries.[1] ABA has been applied in several settings (e.g. to support medical decision-making [7] and e-procurement [8]). ABA's applicability relies on the existence of computational mechanisms, based on various kinds of *dispute derivations* [2, 3, 5] that are formally proven to be correct procedures for conducting structured argumentation under various semantics. For example, [3] proposes *GB-dispute derivations* (GB-DDs in short) for determining whether sentences can be justified by *grounded* sets of arguments and assumptions.

Dispute derivations rely upon the computation of arguments that can be understood as trees [4], in a way similar to other frameworks for structured argumentation (e.g. [9–12]). This introduces both conceptual redundancy and inefficiency in standard methods of implementation, in that within an argument different rules for deriving the same conclusion may be used, potentially introducing unnecessarily points of attack and requiring additional defence efforts.

In this paper, we give a novel computational mechanism for ABA, in the form of *graph-based GB-dispute derivations* (gGB-DDs in short), and prove that

[1] Work was supported by EPSRC grant EP/J020915/1 (TRaDAr—Transparent Rational Decisions by Argumentation) and by an Imperial College London Medical Engineering Kick-Start scheme.

they are correct under the *grounded* semantics for argumentation, in the same way that GB-DDs are. However, gGB-DDs avoid the conceptual redundancy and inefficiency of arguments as trees, by computing *rule-minimal arguments*, corresponding to *argument graphs*, and making sure that only *parsimonious sets of arguments* from the grounded set are generated in support of sentences whose acceptability is being ascertained. In addition to using *argument graphs*, gGB-DDs also incorporate a loop-checking mechanism, inspired by that proposed by [13] for abstract argumentation [6].

In addition to studying theoretical properties of correctness of gGB-DDs, we also perform an empirical evaluation thereof, by comparing it with an implementation of standard GB-DDs. We perform two groups of experiments. The first uses randomly generated frameworks and randomly selected sentences from them, and the second uses a real-life example of the formalization within ABA of treatment recommendations for breast cancer. Both sets of experiments show promise both in terms of completion time for (successful) derivations as well as termination of (unsuccessful) derivations.

2 Background

ABA frameworks [4] are tuples $(\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^{-})$:

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

In the remainder of the paper, we take as given an ABA framework $(\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^{-})$.

In ABA, arguments are proofs using rules and ultimately dependent on assumptions [4]:

- 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 $\top$, where the root is labelled by s and:
  - for all non-leaf nodes N (labelled by s_0), there is some rule $s_0 \leftarrow s_1, \ldots, s_n \in \mathcal{R}$ s.t. either (i) $n = 0$ and the child of N is labelled by $\top$ or (ii) $n > 0$ and N has n children, labelled by $s_1, \ldots, s_n$ respectively; and
  - S is the set of all sentences in $\mathcal{L}$ labelling the leaves;
- an *argument for* $s \in \mathcal{L}$ *supported by* a set of assumptions $A \subseteq \mathcal{A}$ is a proof for s supported by A.

For an argument a for s supported by A, $\mathit{claim}(\mathsf{a}) = s$ (s is the claim of a) and $\mathit{support}(\mathsf{a}) = A$.

In ABA, an argument b *attacks* an argument a iff there is some $a \in \mathit{support}(\mathsf{a})$ s.t. $\bar{a} = s$, where $s = \mathit{claim}(\mathsf{b})$; a set of arguments B attacks a set of arguments A iff some $\mathsf{b} \in \mathsf{B}$ attacks some $\mathsf{a} \in \mathsf{A}$.

There are parallel, equivalent notions for sets of assumptions rather than arguments [3, 14]. *A set of assumptions B attacks a set of assumptions A* iff there is some an argument for $\bar{a}$ supported by some $B' \subseteq B$, for some $a \in A$. Then *a set of assumptions* is deemed:

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

A *sentence* $s \in \mathcal{L}$ is *admissible/complete/grounded* (optionally, *w.r.t.* $A \subseteq \mathcal{A}$) iff there are (i) (respectively) a set of assumptions $A \subseteq \mathcal{A}$ s.t. A is admissible or $A \subseteq A' \subseteq \mathcal{A}$ for some complete/grounded A' and (ii) an argument a s.t. $\mathit{claim}(\mathsf{a}) = s$ and $\mathit{support}(\mathsf{a}) \subseteq A$.

Several algorithms for determining acceptability of sentences in ABA have been proposed (e.g. see [2, 3]). Here, we focus on GB-dispute derivations [3] (GB-DDs in short). Given a *selection function* (taking a multi-set and returning an element occurring in it) a *GB-DD of a defence set* $\Delta \subseteq \mathcal{A}$ for a sentence $s \in \mathcal{L}$ is a finite sequence of tuples[2] $\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0 \rangle, \ldots, \langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n \rangle$ where:
$\mathcal{P}_0 = \{s\}$, $D_0 = \mathcal{A} \cap \{s\}$, $\mathcal{O}_0 = C_0 = \{\}$
$\mathcal{P}_n = \mathcal{O}_n = \{\}$, $\Delta = D_n$
and for every i s.t. $0 \leqslant i < n$, only one σ in $\mathcal{P}_i$ or one S in $\mathcal{O}_i$ is selected, and:

1. If $\sigma \in \mathcal{P}_i$ is selected then
   (i) if $\sigma \in \mathcal{A}$, then
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$ $\quad$ $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{\overline{\sigma}\}\}$
   (ii) if $\sigma \notin \mathcal{A}$, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ s.t. $C_i \cap R = \{\}$ and
   $\mathcal{P}_{i+1} = (\mathcal{P}_i - \{\sigma\}) \cup R$ $\quad$ $D_{i+1} = D_i \cup (\mathcal{A} \cap R)$
2. If S is selected in $\mathcal{O}_i$ and σ is selected in S then
   (i) if $\sigma \in \mathcal{A}$, then
      (a) either σ is ignored, i.e.
      $\mathcal{O}_{i+1} = (\mathcal{O}_i - \{S\}) \cup \{S - \{\sigma\}\}$
      (b) or $\sigma \notin D_i$ and
      $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$ $\quad$ $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$
      $D_{i+1} = D_i \cup (\{\overline{\sigma}\} \cap \mathcal{A})$ $\quad$ $C_{i+1} = C_i \cup \{\sigma\}$
   (ii) if $\sigma \notin \mathcal{A}$, then
   $\mathcal{O}_{i+1} = (\mathcal{O}_i - \{S\}) \cup \big\{(S - \{\sigma\}) \cup R \,\big|\, \sigma \leftarrow R \in \mathcal{R}\big\}$

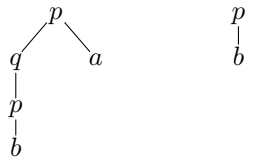
Intuitively, GB-DDs can be seen as games between two (fictitious) players: a *proponent* ($\mathcal{P}_i$) and an *opponent* ($\mathcal{O}_i$), the former accumulating its *supporting/defending assumptions* (D_i), the latter being defeated on a number of *culprit assumptions* (C_i). Theorem 4.2 in [3] proves that GB-DDs are sound: if there is a GB-DD of Δ for s then s is grounded and Δ is admissible and contained in the grounded set of assumptions.

[2] This definition is adapted from [3] but adopting the convention, when defining changes in tuples, that omitted elements are unchanged. We will do the same for gGB-DDs in section 4.

3 Rule-minimal arguments

The notion of argument in ABA enforces a form of relevance (of the support to the claim), afforded by the notion of tree. However, this notion allows redundancies in arguments, in the sense illustrated by the following example. (Note that in depicting arguments as trees and (later) graphs, we follow the convention of letting nodes labelled by the heads of rules appear *above* nodes labelled by the sentences in the rules' bodies; this allows us to omit direction arrows on arcs.)

Example 1. Consider the ABA framework with $\mathcal{R} = \{p \leftarrow b;\quad p \leftarrow q, a;\quad q \leftarrow p\}; \mathcal{A} = \{a, b\}$; $\bar{a} = x$, $\bar{b} = y$. Shown are arguments a_1 (left) and a_2 (right) for p, supported by $\{a, b\}$ and $\{b\}$ respectively. Argument a_1 is (redundantly) using two different rules to prove the two occurrences of p. ⌟



It is clear that such a situation is toxic, in that p has been proved in a certain way, which depends on p itself (which is then proved in a different way). A less toxic sort of case, but still involving redundancy, would be if some sentence s were proved in two different ways in an argument, but without there being one of those proofs depending on the other (so there would be no directed path from the two nodes labelled by s). We take the view that both the 'toxic' and the 'merely redundant' cases are undesirable.

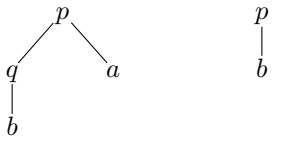
Accordingly, we define a restricted notion of argument, enforcing that, for every sentence in an argument, the same rules are used to justify the sentence at all its occurrences. Formally:

Definition 1. *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* $\mathcal{L} \cup \{\top\}$. ⌟

In example 1, a_2 is rule-minimal whereas a_1 is not. Note that there are infinitely many arguments for p in this example; a_2 is the only rule-minimal one.

Rule-minimal arguments may still contain redundancies in their support, as illustrated by the following example.

Example 2. Consider the ABA framework in example 1 but with $q \leftarrow p$ in $\mathcal{R}$ replaced by $q \leftarrow b$. Shown are rule-minimal arguments a_1 (left) and a_2 (right) for p, supported by $\{a, b\}$ and $\{b\}$ respectively. Here, the support of a_1 is non-minimal, as $\mathit{support}(\mathsf{a}_1) \subset \mathit{support}(\mathsf{a}_2)$. ⌟

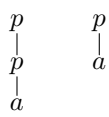


Definition 2. *An argument* a *is* support-minimal *iff there is no* a' *such that* $\mathit{claim}(\mathsf{a}')=\mathit{claim}(\mathsf{a})$ *and* $\mathit{support}(\mathsf{a}')\subset \mathit{support}(\mathsf{a})$. ⌟

Example 2 shows that rule-minimal arguments may not be support-minimal. The following example shows that support-minimal arguments may not be rule-

minimal.

Example 3. Consider the ABA framework with $\mathcal{R} = \{p \leftarrow p; p \leftarrow a\}; \mathcal{A} = \{a\}; \bar{a} = x$. Shown are support-minimal arguments a_1 (left) and a_2 (right) for p, both supported by $\{a\}$. Here, only a_2 is rule-minimal. (This is the smallest such example; other, less 'trivial' ones could be provided.) ⌟



Whereas the notion of support-minimal argument is 'global', in that to check whether an argument is support-minimal this needs to be compared with all other arguments, the notion of rule-minimal argument is 'local', in that to check whether an argument is rule-minimal all that is required is a syntactic check of the argument. Moreover, every argument can be transformed into a rule-minimal argument, by means of algorithm 1.[3]

Algorithm 1 *reduce*(a: argument)

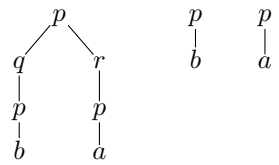
```
1:  r := 0
2:  seen := {}
3:  while r ⩽ rank(a) do
4:     nodes := {N ∈ nodes(a) | (rank(N, a) = r) ∧ (label(N, a) ∈ ℒ − 𝒜) ∧ (N ∉ seen)}
5:     while nodes ≠ {} do
6:        N := pickOne(nodes)
7:        s := label(N, a)
8:        leafTrees := {b ∈ subTrees(a)|(label(root(b)) = s) ∧ ¬∃N'[N' ∈ nodes(b) −
          {root(b)} ∧ label(N', a)=s]}
9:        b := pickOne(leafTrees)
10:       for all N' ∈ nodes(a) s.t. label(N', a) = s ∧ ¬∃X[X ∈ path(N', root(a), a) ∧
          label(X) = s] do
11:          a := substitute(a, N', b)
12:       end for
13:       seen := seen ∪ {N ∈ nodes|label(N) = s}
14:       nodes := nodes − seen
15:    end while
16:    r := r + 1
17: end while
18: return a
```

Note that at lines 6 and 9 the algorithm performs non-deterministic choices (of a node/sentence and of a sub-tree, respectively). By making alternative such

[3] 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 sub-tree 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 sub-trees 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 section 2).

choices different arguments can be obtained, as illustrated next.

Example 4. Given argument a_1 (left), depending on the choice of sub-tree at line 9, the algorithm may return a_2 (middle) or a_3 (right). ⌟



Definition 3. *Given a set of arguments* A*, a* reduction *of* A *is a set of arguments* B *s.t. (i) for each* $\mathsf{b} \in \mathsf{B}$ *there is an argument* $\mathsf{a} \in \mathsf{A}$ *s.t.* $\mathsf{b} = \mathit{reduce}(\mathsf{a})$*; (ii) for each argument* $\mathsf{a} \in \mathsf{A}$ *there exists an argument* $\mathsf{b} \in \mathsf{B}$ *s.t.* $\mathsf{b} = \mathit{reduce}(\mathsf{a})$*.* ⌟

In example 4, $\{\mathsf{a}_2\}$, $\{\mathsf{a}_3\}$, $\{\mathsf{a}_2,\mathsf{a}_3\}$ are reductions of $\{\mathsf{a}_1\}$.

In general, given an argument a, algorithm 1 'reduces' it to a rule-minimal argument a' whose claim is identical to, and whose support is a subset of that of a, as sanctioned by:

Proposition 1. *Let* a *be an argument for c supported by S. Then* $\mathsf{a}' = \mathit{reduce}(\mathsf{a})$ *is a rule-minimal argument for c supported by* $S' \subseteq S$*.*

Proof. First, algorithm 1 terminates, since (i) a *is finite and thus rank*(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 sub-tree of* a *with the same root, and so the same claim and support as* a*. Thirdly, trivially* a' *is an argument in the ABA sense. Finally, by construction, each sentence in* a' *is proven by only one rule.* ⌟

Then, directly from proposition 1:

Proposition 2. *(i) For every rule-minimal argument for s supported by A there exists an argument of s supported by A. (ii) For every argument for s supported by A there exists a rule-minimal argument of s supported by* $A' \subseteq A$*.* ⌟

Clearly, it is computationally advantageous, when determining whether a sentence is acceptable under some semantics, to focus on rule-minimal arguments: there are fewer of them, they are smaller, and they have smaller supports. We define new notions of acceptability w.r.t. rule-minimal arguments, and prove they are equivalent to the original notions.

To extend notions of acceptability for sets of assumptions when focusing on rule-minimal arguments, we define a variant of the notion of attack between sets of assumptions:

Definition 4. *A set of assumptions B* rule-minimally attacks *a set of assumptions A iff there is some rule-minimal argument for* $\bar{a}$ *supported by some* $B' \subseteq B$*, for some* $a \in A$*.*

A set of assumptions *is*

- rule-minimally admissible *iff it does not rule-minimally attack itself and it rule-minimally attacks every set of assumptions rule-minimally attacking it;*

– rule-minimally complete *iff it is rule-minimally admissible and contains all assumptions it can defend (by rule-minim. attacking all rule-min. attacks against them);*
– rule-minimally grounded *iff it is minimally (w.r.t. $\subseteq$) rule-minimally complete.* ⌟

Directly from proposition 2:

Proposition 3. *Let $A \subseteq \mathcal{A}$ be a set of assumptions. A is admissible/complete/grounded iff A is rule-minimally admissible/complete/grounded (respectively).* ⌟

Thus, when deciding whether a set of assumptions is acceptable, one can restrict attention to rule-minimal arguments.

As in the case of standard ABA, we can lift notions of acceptability at the assumption level to the sentence level:

Definition 5. *$s \in \mathcal{L}$ is* rule-minimally admissible/complete/grounded *(optionally, w.r.t. $A \subseteq \mathcal{A}$) iff there are (i) (respectively) a set of assumptions $A \subseteq \mathcal{A}$ s.t. A is rule-minimally admissible or $A \subseteq A' \subseteq \mathcal{A}$ for some rule-minimally complete/grounded A' and (ii) a rule-minimal argument a s.t. claim$(\mathsf{a}) = s$ and support$(\mathsf{a}) \subseteq A$.* ⌟

Then, directly from proposition 3:

Proposition 4. *$s \in \mathcal{L}$ is rule-minimally admissible/complete/grounded/iff s is admissible/complete/grounded (respectively).* ⌟

Rule-minimally acceptable sets of assumptions may still contain redundancies, as illustrated by the following example.

Example 5. Consider the ABA framework with $\mathcal{R} = \{(1)\ p \leftarrow s, r, a; (2)\ s \leftarrow r, a; (3)\ r \leftarrow a; (4)\ p \leftarrow b; (5)\ q \leftarrow d;\ (6)\ q \leftarrow e\}$; $\mathcal{A} = \{a, b, c, d, e\}$; $\bar{a} = x, \bar{b} = y, \bar{c} = q, \bar{d} = p, \bar{e} = p$. (The rules are numbered for later use.) Then c is (rule-minimally) admissible, complete and grounded, w.r.t. $\{c, a, b\}$, $\{c, a\}$ and $\{c, b\}$. $\{c, a\}$ and $\{c, b\}$ determine a more parsimonious set of arguments, in that p is supported by $\{a\}$ (in the case of $\{c, a\}$) or $\{b\}$ (for $\{c, b\}$). ⌟

Definition 6. *A set of arguments A is* parsimonious *iff there exist no two different sub-trees a, b of any (possibly different) arguments in A such that the root of a and b is labelled by the same sentence.* ⌟

Every argument in a parsimonious set is rule-minimal. In example 5, the set of assumptions $\{c, a\}$ and $\{c, b\}$ support parsimonious arguments, whereas $\{c, a, b\}$ does not.

It is easy to see that in order to determine acceptability of sentences, it suffices to focus on parsimonious arguments:

Proposition 5. *A sentence s is rule-minimally admissible/complete/grounded iff there are (i) a parsimonious set of arguments A and (ii) $\mathsf{a} \in \mathsf{A}$ with claim$(\mathsf{a}) = s$ s.t. (respectively) A is admissible or $\mathsf{A} \subseteq \mathsf{A}'$ for some complete/grounded A'.* ⌟

The relevance of this will be seen in the following section. When constructing a set A of proponent arguments (according to the algorithm in Definition 8) starting from some claim s, we can restrict attention to parsimonious A; this is a further efficiency and removal of redundancy.

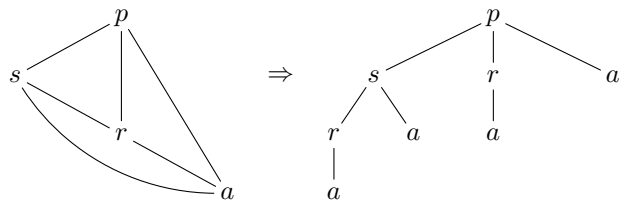
4 Graph-based GB-dispute derivations

A graph-based GB dispute derivation gradually derives justifications for sentences in a way guaranteed to produce rule-minimal arguments which are parsimonious and grounded. They rely upon arguments defined as graphs, as follows:

Definition 7. *A* graph-based argument *is an acyclic directed graph* (V, E) *with* $V \subseteq (\mathcal{L} \cup \{\top\})$*, and for any* $s \in V$*:*

- *if* $s \in (\mathcal{L} - (\mathcal{A} \cup \{\top\}))$*, then for a unique rule* $s \leftarrow s_1, \ldots, s_m$ *in* $\mathcal{R}$*, (i) if* $n = 0$*, then* $\{x | (s, x) \in E\} = \{\top\}$*; or (ii) if* $n > 0$*, then* $\{x | (s, x) \in E\} = \{s_1, \ldots, s_n\}$*;*
- *if* $s \in V - (\mathcal{L} - (\mathcal{A} \cup \{\top\}))$*, then there are no outgoing edges from* s *in* E*;*
- *there is a unique* $c \in (V \cap \mathcal{L})$ *(the* claim*) s.t. there is no edge* (s, c) *in* E *and there is a path* $(c, \ldots, s)$ *for any* $s \in V$*.* ⌟

It is evident that argument graphs can be 'unravelled' into rule-minimal arguments, as illustrated below for example 5:



gGB-DDs work over tuples $(\mathcal{P}_i, \mathcal{O}_i, D_i, C_i, JsP_i, JsO_i, Att_i, G_i)$, where $\mathcal{P}_i$ are the sentences the proponent has yet to prove; $\mathcal{O}_i$ contains tuples $(X, Js, \mathbf{C})$ representing partially-completed opponent arguments: $\mathbf{C}$ is the claim, X the sentences supporting the argument yet to be proved, and Js a set of *justifications*–pairs (s, R) where s is a sentence and R is either the body of a rule used to justify s in the context of $(X, Js, \mathbf{C})$, or $*$ if $s \in \mathcal{A}$; D_i and C_i are as in GB-dispute derivations (see section 2); JsP_i contains justifications (pairs (s, R), as above) for the proponent arguments, and JsO contains the justification triples for the opponent arguments; Att_i contains points of attack between proponent and opponent arguments, and G_i records the dependency graph among sentences, grown gradually during the derivation.

Definition 8. *Given a selection function, a* gGB-DD of defence set Δ and dialectical structure (JsP, JsO, Att) *for a sentence* $s \in \mathcal{L}$ *is a finite sequence of tuples* $(\mathcal{P}_0, \mathcal{O}_0, D_0, C_0, JsP_0, JsO_0, Att_0, G_0), \ldots, (\mathcal{P}_n, \mathcal{O}_n, D_n, C_n, JsP_n, JsO_n, Att_n, G_n)$*, where*

$$\mathcal{P}_0 = \{s\}, D_0 = \mathcal{A} \cap \{s\}, \mathcal{O}_0 = C_0 = JsP_0 = JsO_0 = Att_0 = G_0 = \{\}$$
$$\mathcal{P}_n = \mathcal{O}_n = \{\}, \Delta = D_n, JsP = JsP_n, JsO = JsO_n, Att = Att_n$$

and for every i s.t. $0 \leqslant i < n$, only one σ in $\mathcal{P}_i$ or one $(X, Js, \boldsymbol{C})$ in $\mathcal{O}_i$ is selected, and:

1. *If $\sigma \in \mathcal{P}_i$ is selected then*
   (i) if $\sigma \in \mathcal{A}$ then

$$\begin{aligned}
\mathcal{P}_{i+1} &= \mathcal{P}_i - \{\sigma\} \\
\mathcal{O}_{i+1} &= \mathcal{O}_i \cup \{(\{\bar{\sigma}\}, \{\}, \bar{\sigma}) \mid \neg\exists R((\bar{\sigma}, R) \in JsO_i\} \\
JsP_{i+1} &= JsP_i \cup \{(\sigma, *)\} \\
Att_{i+1} &= Att_i \cup \{(\bar{\sigma}, \sigma)\} \\
G_{i+1} &= G_i \cup \{(\bar{\sigma}, \sigma)\}, \text{ and } G_{i+1} \text{ is acyclic}
\end{aligned}$$

   (ii) if $\sigma \notin \mathcal{A}$, then (a) there is some $(\sigma, R) \in JsP_i$, and $\mathcal{P}_{new}$ is $\{\}$; or, if not, (b) there exists some $\sigma \leftarrow R \in \mathcal{R}$, $\mathcal{P}_{new}$ is R—and (in both cases) $C_i \cap R = \{\}$ and

$$\begin{aligned}
\mathcal{P}_{i+1} &= (\mathcal{P}_i - \{\sigma\}) \cup \mathcal{P}_{new} \\
D_{i+1} &= D_i \cup (R \cap \mathcal{A}) \\
JsP_{i+1} &= JsP_{i+1} \cup \{(\sigma, R)\} \\
G_{i+1} &= G_i \cup \{(x, \sigma) \mid x \in R\}, \text{ and } G_{i+1} \text{ is acyclic}
\end{aligned}$$

2. *If $(X, Js, \boldsymbol{C})$ is selected in $\mathcal{O}_i$ and σ is selected in X then*
   (i) if $\sigma \in \mathcal{A}$, then:
      (a) σ is ignored, i.e. $\mathcal{O}_{i+1} = (\mathcal{O}_i - \{(X, Js, \boldsymbol{C})\}) \cup \{(X - \{\sigma\}, Js \cup \{(\sigma, *)\}, \boldsymbol{C})\}$.
      (b) or $\sigma \notin D_i$ and if $\exists R((\bar{\sigma}, R) \in JsP_i)$ then $P_{new} = \{\}$; otherwise, $P_{new} = \{\bar{\sigma}\}$ and

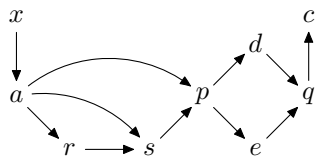
$$\begin{aligned}
\mathcal{P}_{i+1} &= \mathcal{P}_i \cup P_{new} \\
\mathcal{O}_{i+1} &= \mathcal{O}_i - \{(X, Js, \boldsymbol{C})\}) \\
D_{i+1} &= D_i \cup (\{\bar{\sigma}\} \cap \mathcal{A}) \\
C_{i+1} &= C_i \cup \{\sigma\} \\
JsO_{i+1} &= JsO_i \cup \{(X - \{\sigma\}, Js \cup \{(\sigma, *)\}, \boldsymbol{C})\} \\
Att_{i+1} &= Att_i \cup \{(\bar{\sigma}, \sigma)\} \\
G_{i+1} &= G_i \cup \{(\bar{\sigma}, \sigma)\}, \text{ and } G_{i+1} \text{ is acyclic}
\end{aligned}$$

   (ii) if $\sigma \notin \mathcal{A}$ then
      – if $\exists R((\sigma, R) \in Js)$, let $O_{new} = \{((X - \{\sigma\}) \cup R, Js, \boldsymbol{C})\}$ and let $G_i = G_i$;*
      – otherwise let $\mathcal{O}_{new} = \{((X - \{\sigma\}) \cup R, Js \cup \{(\sigma, R)\}, \boldsymbol{C}) \mid (\sigma \leftarrow R) \in \mathcal{R}\}$ and let $G_i* = G_i \cup \{(x, \sigma) \mid \exists(\sigma \leftarrow R) \in \mathcal{R},\ x \in R\}$ and G_i* is acyclic.
   then: $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \mathcal{O}_{new}$ and $G_{i+1} = G_i*$. ⌟

	$\mathcal{P}$	$\mathcal{O}$	D	C	JsP	JsO
0	$\{c\}$	$\{\}$	$\{c\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{\}$	$\{(\{q\},\{\},q)\}$	$\{c\}$	$\{\}$	$\{(c,*)\}$	$\{\}$
2	$\{\}$	$\{(\{d\},\{(q,4)\},q),$ $(\{e\},\{(q,5)\},q)\}$	$\{c\}$	$\{\}$	$\{(c,*)\}$	$\{\}$
3	$\{p\}$	$\{(\{e\},\{(q,5)\},q)\}$	$\{c\}$	$\{d\}$	$\{(c,*)\}$	$\{(\{\},\{(d,*),(q,4)\},q)\}$
4	$\{s,a\}$	$\{(\{e\},\{(q,5)\},q)\}$	$\{a,c\}$	$\{d\}$	$\{(c,*),(p,1)\}$	$\{(\{\},\{(d,*),(q,4)\},q)\}$
5	$\{a,r\}$	$\{(\{e\},\{(q,5)\},q)\}$	$\{a,c\}$	$\{d\}$	$\{(c,*),(p,1),(s,2)\}$	$\{(\{\},\{(d,*),(q,4)\},q)\}$
6	$\{a\}$	$\{(\{e\},\{(q,5)\},q)\}$	$\{a,c\}$	$\{d\}$	$\{(c,*),(p,1),(r,3),(s,2)\}$	$\{(\{\},\{(d,*),(q,4)\},q)\}$
7	$\{\}$	$\{(\{e\},\{(q,5)\},q),$ $(\{x\},\{\},x)\}$	$\{a,c\}$	$\{d\}$	$\{(a,*),(c,*),(p,1),$ $(r,3),(s,2)\}$	$\{(\{\},\{(d,*),(q,4)\},q)\}$
8	$\{\}$	$\{(\{x\},\{\},x)\}$	$\{a,c\}$	$\{d,e\}$	$\{(a,*),(c,*),(p,1),(r,3),(s,2)\}$	$\{(\{\},\{(d,*),(q,4)\},q),(\{\},\{(e,*),(q,5)\},q)\}$
9	$\{\}$	$\{\}$	$\{a,c\}$	$\{d,e\}$	$\{(a,*),(c,*),(p,1),(r,3),(s,2)\}$	$\{(\{\},\{(d,*),(q,4)\},q),(\{\},\{(e,*),(q,5)\},q)\}$

Table 1. Sample gGB-DD for c in example 5.

As an illustration, consider table 1 (Att_i and G_i are omitted for lack of space). The opponent has two arguments attacking the claim c, introduced in step 2 when the incomplete argument for q was developed using rules (4) and (5) (in example 5). The proponent attacks opponent argument $(\{d\},\{(q,4)\},q)$ using rule (1) for p (step 4). Then, when the proponent must attack the second opponent argument $(\{e\},\{(q,5)\},q)$, at step 8, the algorithm notices that $\bar{e}=p$ has already been argued for by the proponent (at case 2(i)(b) in definition 8, the condition which sets P_{new} to $\{\}$), so an argument for p is not developed again (avoiding the possibility that it would be developed using an alternative rule). It is here that we ensure parsimoniousness. The acyclicity check on G_i ensures that this avoidance of recomputation is sound; the final graph G_n is shown below.

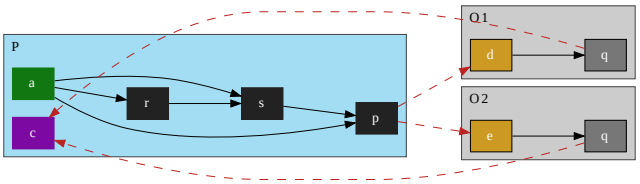


Definition 9. *Let J be a set of pairs of the form (s,R); if $s\in\mathcal{A}$ then R is $*$; otherwise there exists some rule $s\leftarrow R\in\mathcal{R}$. The* arguments determined by J *are those constructible from the ABA framework $(\mathcal{L}',\mathcal{R}',\mathcal{A}',{}^{-\prime})$:*

$\mathcal{L}'=\{s\mid\exists(s',R)\in J[s=s'\vee(R\neq *\wedge s\in R)]\}$;
$\mathcal{R}'=\{s\leftarrow R\mid(s,R)\in J,\ R\neq *\}$;
$\mathcal{A}'=\{a\mid(a,*)\in J\}$; $\quad\bar{a}'=a$, *for all* $a\in\mathcal{A}$. ⌟

It is apparent that where, for any s, there is at most one pair (s,R) in JsP, then the set of arguments determined by JsP is parsimonious. Furthermore, the set JsP can be more compactly visualized as a graph, whose nodes are the sentences mentioned in JsP, and where there is an edge (s,r) iff there is a pair $(s,R)\in JsP$

s.t. $r \in R$. For the gGB-DD in table 1, this visualization is shown below, together with the dialectical relationship with opponent arguments from JsO:



In the diagram, the proponent's justifications are shown in the large box on the left; the opponent's arguments are the two small boxes on the right; and attacks are dashed lines.

Proposition 6. *If there is a gGB-DD for s then there is a GB-DD for s with the same defence set.*

Proof. (Sketch: the details are omitted for reasons of space.) The structure of gGB-DDs precisely mirrors that presented in section 2 for GB-DDs; the sets $\mathcal{P}_i$ are the same, and the members $(X, Js, \boldsymbol{C})$ of the sets $\mathcal{O}_i$ have components X which precisely correspond to the members of $\mathcal{O}_i$ in GB-DDs. However, because of the checks at steps 1(ii), 2(i)(b) and 2(ii) of gGB-DDs, some steps of a GB-DD may be omitted in a gGB-DD. So, given a gGB-DD $(\mathcal{P}_0, \mathcal{O}_0, D_0, C_0, JsP_0, JsO_0, Att_0, G_0), \ldots, (\mathcal{P}_m, \mathcal{O}_m, D_m, C_m, JsP_m, JsO_m, Att_m, G_m)$ there is a GB-DD $(\mathcal{P}'_0, \mathcal{O}'_0, D'_0, C'_0), \ldots, (\mathcal{P}'_n, \mathcal{O}'_n, D'_n, C'_n)$ with $(m \leqslant n)$, such that to each step $(\mathcal{P}_i, \mathcal{O}_i, D_i, C_i, JsP_i, JsO_i, Att_i, G_i)$ there corresponds a step of the GB-DD $(\mathcal{P}'_j, \mathcal{O}'_j, D'_j, C'_j)$ with $\mathcal{P}'_j = \mathcal{P}$, $\mathcal{O}'_j = \{X \mid \exists (X, Js, \boldsymbol{C}) \in \mathcal{O}_i\}$, $D'_j = D_i$, $C'_j = C_i$– and s.t. the corresponding steps fall into the same order. ⌟

The table below shows the GB-DD corresponding to the gGB-DD of table 1. The numbers of corresponding steps from the gGB-DD are in brackets.

	$\mathcal{P}$	$\mathcal{O}$	D	C
0 [0]	$\{c\}$	$\{\}$	$\{c\}$	$\{\}$
1 [1]	$\{\}$	$\{\{q\}\}$	$\{c\}$	$\{\}$
2 [2]	$\{\}$	$\{\{d\},\{e\}\}$	$\{c\}$	$\{\}$
3 [3]	$\{p\}$	$\{\{e\}\}$	$\{c\}$	$\{d\}$
4 [4]	$\{s,a\}$	$\{\{e\}\}$	$\{a,c\}$	$\{d\}$
5 [5]	$\{a,r\}$	$\{\{e\}\}$	$\{a,c\}$	$\{d\}$
6 [6]	$\{a\}$	$\{\{e\}\}$	$\{a,c\}$	$\{d\}$
7 [7]	$\{\}$	$\{\{e\},\{x\}\}$	$\{a,c\}$	$\{d\}$
8	$\{p\}$	$\{\{x\}\}$	$\{a,c\}$	$\{d,e\}$
9	$\{s,a\}$	$\{\{x\}\}$	$\{a,c\}$	$\{d,e\}$
10	$\{a,r\}$	$\{\{x\}\}$	$\{a,c\}$	$\{d,e\}$
11	$\{a\}$	$\{\{x\}\}$	$\{a,c\}$	$\{d,e\}$
12 [8]	$\{\}$	$\{\{x\}\}$	$\{a,c\}$	$\{d,e\}$
13 [9]	$\{\}$	$\{\}$	$\{a,c\}$	$\{d,e\}$

Proposition 6 also holds in the reverse, 'completeness' direction, with the modification that the defence set may be a subset of that for the corresponding GB-DD.

Proposition 7. *If there is a gGB-DD for s with defence set Δ, then s is grounded, Δ is admissible, and there is $\Delta' \supseteq \Delta$ s.t. Δ' is grounded.*

Proof. By proposition 6, there is a GB-DD for s with defence set Δ; then from Theorem 4.2 of [3], the result is immediate. ⌟

Proposition 8. *If there is a gGB-DD for s with defence Δ, then the arguments determined by JsP are parsimonious.*

Proof. By construction, there are no two pairs (s, R_1), (s, R_2) in JsP with $R_1 \neq R_2$. Thus the arguments determined by JsP can only be parsimonious. ⌟

5 Experiments

To compare the original GB-DDs to the gGB-DDs of section 4, we implemented both in Prolog. The implementation of the original algorithm (`proxdd`) used its (equivalent) variant presented in [5], 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 (`grapharg`) record the rule-minimal justification structure as it proceeds.[4]

In comparing the results of the two implementations, it is important to set the same search strategy in each case. Each algorithm has various choice points (indicated by words such as 'selected', or disjunctions), and to compare like with like it is necessary that the selection be done using the same criteria.

Another fact we had to consider was that, for many strategies, the original GB-DDs quickly used up all of Prolog's memory resources. (For such strategies, the gGB-DD implementation, `grapharg`, typically terminated or timed out.) We therefore used strategies for which memory was typically not exceeded for both implementations.

For the first set of experiments, we randomly generated ABA frameworks $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ })$ to use as experimental data. The random generation followed a very simple procedure of choosing contraries to assumptions at random, and populating rule bodies with sentences randomly. The proportions of assumptions to non-assumptions in the language; the minimum and maximum number of sentences per rule body; the minimum and maximim number of rules per sentence serving as rule head—these and similar parameters can all be supplied by the user. In our experiments, the mean language size ($|\mathcal{L}|$) was 126 sentences, and the mean number of rules ($|\mathcal{R}|$) was 178, with a mean of 3.6 sentences in the body of each rule. For each randomly-selected framework, we randomly selected 10 sentences from the language of the framework to use as queries. We tested

[4] Both implementations are freely available for download from `http://www.doc.ic.ac.uk/~rac101/proarg/`.

each query-framework pair for both implementations, on the same strategy, with a time-out of two minutes. In all cases we asked the implementations to find all possible solutions. The results, for 65 frameworks, are presented in tabular form below (times in secs.):

	`grapharg`	`proxdd`
Av. time (both complete)	0.447	6.618
Av. time (overall)	54.716	87.413
percentage timeout	40.871	71.208

We recorded the mean time for query-framework pairs where both implementations completed (first row), as well as the mean time for query-framework pairs that may have reached the chosen time-out of two minutes.

Our algorithm shows a marked improvement in the mean times taken to answer queries, in the two cases where both implementations completed, and when one of them reached time-out. The percentage of time-outs itself was much lower for gGB-DDs. These results are encouraging and confirm our theoretical evaluation.

We were surprised by the comparisons on number of solutions found: in the cases where both implementations completed, the same number of solutions were discovered. One might have expected that the guarantee of rule-minimality in the case of the graph-based algorithm would have meant that fewer solutions would have been produced by the graph-based algorithm, with the non-rule-minimal ones being cut. The fact that the figures are the same in each case is an indication that the ABA frameworks our random-generator produced did not exhibit the scope for non-rule-minimal arguments—for the chosen queries, at least. Finally, the very high number of total solutions found overall for the graph-based algorithm (13,653 vs 217) is owed to one particular randomly-generated query-framework pair, for which `grapharg` found 13,427 solutions (proxdd found none before time-out). If that particular query-framework pair is left out of account, then the comparison comes to 226 vs 217.

For the experiments based on the breast-cancer study, we used data originally published in [15], and which we have used in the context of experiments on parallel argumentation in [7]. The ABA frameworks represent an ontology of drugs and treatments, and recommendations from 57 papers referred to in the National Cancer Institute's breast cancer guideline [16], as well as hypothetical patient data. In half of the frameworks, we introduced random preferences over the recommendations from the various clinical trials; this simulates the weights which patients or doctors might give to the various clinical trials from which the recommendations are drawn. Further, in half of the frameworks, we flattened the ontology to a set of Prolog facts, rather than retaining the original combination of Prolog facts and rules. The ABA resulting frameworks consist of an average of 947 rules ($|\mathcal{R}|$), and 11 queries were made per framework—each query asking whether a particular regime of chemotherapy or drugs was recommended.

We again made these experiments using our graph-based implementation, `grapharg`, and compared it to the existing best implementation of the standard algorithm, `proxdd`. The results are shown in tabular form below.

	grapharg	proxdd
Av. time (both complete)	0.575	3.827
Av. time (overall)	23.630	46.015
percentage timeout	18.182	36.364

The results here are broadly consistent with those obtained for the randomly-generated frameworks in the previous round of experiments, and indicate a similar, increased performance and utility in the case of graph-based dispute derivations.

6 Conclusion

We proposed an equivalent but 'leaner' form of ABA, based on rule-minimal, graph-based arguments, and gave a variant of an existing mechanism for computation under the grounded semantics in ABA, namely GB-dispute derivations [3] (GB-DDs in short), to restrict computation to graph-based arguments only. We have proven theoretically that our graph-based GB-dispute derivations (gGB-DDs in short) are sound, under the grounded semantics, and conducted a number of experiments suggesting that our gGB-DDs are more efficient than standard GB-DDs, both in terms of completion time and terminating computations.

Like others, e.g. [11], we are concerned with 'efficient' arguments, but, rather than imposing minimality of support of arguments, which needs to be ascertained 'globally', we propose *rule-minimality* for arguments, which can be ascertained 'locally'. Our notion of rule-minimality is related to the definition of argument structure (Def. 3.1) in [10].

Our notion of gGB-DD borrows from the work of [13] the use of a graph whose acyclicity is an essential prerequisite of success. However, whereas [13] provide a computational machinery for abstract argumentation [6], we have focused on structured argumentation in the form of ABA. Moreover, [13] consider several argumentation semantics; we have focused on the grounded semantics.

There is an established completeness result for the derivation algorithm for GB-DDs, from [3], which holds in the case of p-acyclic ABA frameworks. This result is inherited for the algorithms defined in the present paper, and it is straightforward to show the inheritance. We have omitted this for reasons of space in the current paper.

In future work, it would be interesting to see whether our notions of rule-minimal and graph-based arguments could be applied in other frameworks for structured argumentation, e.g. those of [9–12].

We conducted preliminary experimentation with an implementation of our gGB-DDs and shown that it moderately, but consistently, improves upon an implementation of standard GB-DDs. We plan to further this experimentation to a larger pool of frameworks and queries.

Like us, [7] also focus on obtaining more efficient computational support for ABA in the context of a medical application, but by resorting to parallelisation, where different strategies lead to different threads of execution. It would be interesting to see how parallelisation could further quicken our implementation.

We have focused on the computation of argumentation under the grounded semantics. We have already defined variants of our gGB-DDs to compute the admissible semantics, and implemented that in `grapharg`. We plan also to define and implement a variant for the ideal semantics; using the parametric methodology of [5] this should be straightforward.

References

1. Bondarenko, A., Dung, P.M., Kowalski, R., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence **93**(1-2) (1997) 63–101
2. Dung, P., Kowalski, R., Toni, F.: Dialectic proof procedures for assumption-based, admissible argumentation. Artificial Intelligence **170** (2006) 114–159
3. Dung, P., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artificial Intellgence **171**(10–15) (2007) 642–674
4. Dung, P.M., Kowalski, R.A., Toni, F.: Assumption-based argumentation. In Rahwan, I., Simari, G.R., eds.: Argumentation in AI. Springer (2009) 25–44
5. Toni, F.: A generalised framework for dispute derivations in assumption-based argumentation. Artificial Intelligence (2012) In Press.
6. Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77** (1995) 321–357
7. Craven, R., Toni, F., Hadad, A., Cadar, C., Williams, M.: Efficient support for medical argumentation. In Brewka, G., Eiter, T., McIlraith, S.A., eds.: Proc. 13th International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press (2012) 598–602
8. Matt, P.A., Toni, F., Stournaras, T., Dimitrelos, D.: Argumentation-based agents for eprocurement. In Berger, M., Burg, B., Nishiyama, S., eds.: Proceedings of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)-Industry and Applications Track. (2008) 71–74
9. Modgil, S., Prakken, H.: A general account of argumentation with preferences. Artificial Intelligence (2012) In Press.
10. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming **4**(1-2) (2004) 95–138
11. Besnard, P., Hunter, A.: Elements of Argumentation. MIT Press (2008)
12. Amgoud, L.: The outcomes of logic-based argumentation systems under preferred semantics. In Hüllermeier, E., Link, S., Fober, T., Seeger, B., eds.: Scalable Uncertainty Management - 6th International Conference, SUM 2012. Proceedings. Volume 7520 of LNCS., Springer (2012) 72–84
13. Thang, P.M., Dung, P.M., Hung, N.D.: Towards a common framework for dialectical proof procedures in abstract argumentation. Journal of Logic and Computation **19**(6) (2009) 1071–1109
14. Toni, F.: Reasoning on the web with assumption-based argumentation. In Eiter, T., Krennwallner, T., eds.: Reasoning Web. Semantic Technologies for Advanced Query Answering. 8th International Summer School. Proceedings. Volume 7487 of LNCS. Springer (2012) 370–386
15. Williams, M., Hunter, A.: Harnessing Ontologies for Argument-Based Decision-Making in Breast Cancer. In: ICTAI (2), IEEE Computer Society (2007) 254–261
16. NCI: Breast Cancer PDQ (Stage I, II, IIA, and operable IIIC Breast Cancer) (2007)