Hybrid argumentation and its properties

Dorian GAERTNER and Francesca TONI
Department of Computing, Imperial College London, UK

Abstract. We present a variant of AB-dispute derivations for assumption-based argumentation (ABA), that can be used for determining the admissibility of claims. ABA reduces the problem of computing arguments to the problem of computing assumptions supporting these arguments. Whereas the original AB-dispute derivations only manipulate sets of assumptions, our variant also renders explicit the underlying dialectical structure of arguments (by a proponent) and counter-arguments (by an opponent), and thus supports a hybrid of ABA and abstract argumentation beneficial to developing applications of argumentation where explicit justifications of claims in terms of full dialectical structures are required. We prove that the proposed variant of AB-dispute derivations is correct.

Keywords. Abstract Argumentation, Assumption-based Argumentation, Computation

1. Introduction

Argumentation has proved to be a useful abstraction mechanism for understanding several problems, and a number of computational frameworks for argumentation have been proposed in order to provide tools to address these problems. These frameworks are mostly based upon abstract argumentation [6], that focuses on determining the admissibility of arguments based upon their capability to counter-attack all arguments attacking them, while being conflict-free. In abstract argumentation the *arguments* and the *attack* relation between arguments are seen as primitive notions. The abstract view of argumentation is equipped with intuitive and simple computational models (e.g. [4,14]), but does not allow to address the problems of (i) how to find arguments, (ii) how to determine attacks and (iii) how to exploit the fact that different arguments may share premises. Assumption-based argumentation [1,7,9] is a general-purpose framework for argumentation, where arguments, rather than being a primitive concept, are defined as *backward deductions* (using sets of *rules* in an underlying logic) supported by sets of *assumptions*, and the notion of attack amongst arguments is reduced to that of *contrary* of assumptions. Intuitively, assumptions are sentences that can be assumed to hold but can be questioned and disputed (as opposed to axioms that are instead beyond dispute), and the contrary of an assumption stands for the reason why that assumption may be undermined and thus may need to be dropped.

Existing computational models for assumption-based argumentation [7,9] allow to determine the "acceptability" of claims under the semantics of credulous, admissible extensions as well as under two sceptical semantics (of grounded and ideal extensions). In this paper, we focus on the computational model for admissibility, called AB-dispute derivations [7,9]. These derivations can be seen as a game between two (fictional) players

– a *proponent* and an *opponent* – with rules roughly as follows: the opponent can dispute the proponent's arguments by attacking one of the arguments' supporting assumptions; the proponent can in turn defend its arguments by counter-attacking the opponent's attacks with other arguments, possibly with the aid of other defending assumptions; the proponent does not need to counter-attack any assumption it has already attacked previously or defend any assumption is has already defended previously; the proponent cannot attack any of its own assumptions. While conducting this game, the players explore implicitly a dialectical structure of arguments by the proponent, counter-arguments by the opponent, arguments by the proponent attacking the counter-arguments and so on. However, while doing so, AB-dispute derivations only keep track of the assumptions underlying these arguments, and the dialectical structure is lost.

In this paper, we define generalised structured AB-dispute derivations, a variant of AB-dispute derivations computing explicitly the dialectical structure hidden in AB-dispute derivations and thus providing a hybrid ABA-abstract argumentation mechanisms. Our structured AB-dispute derivations are defined for a generalisation of ABA allowing for *multiple contraries*. Also, they are a (non-trivial) generalisation of the structured AB-dispute derivations defined in [11]: whereas those relied upon a special *patient* selection function for exploring and building arguments, we do not commit to any selection function (or other design choice). Moreover, whereas in [11] we were concerned with the implementation of structured AB-dispute derivations, here we are concerned with formal proofs of correctness (missing in [11]).

The paper is organised as follows: in Section 2 we present the background on ABA and existing notions of AB-dispute derivations. In Section 3 we detail our novel generalised structured AB-dispute derivations. We prove correctness results in Section 4 and conclude in Section 5.

2. Assumption-based argumentation

This section provides the basic background on assumption-based argumentation (ABA), see [1,7,9] for details. An ABA framework is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ where

- $(\mathcal{L}, \mathcal{R})$ is a *deductive system*, consisting of a language $\mathcal{L}$ and a set $\mathcal{R}$ of inference rules,
- $\mathcal{A} \subseteq \mathcal{L}$, referred to as the set of *assumptions*,
- $\overline{\ }$ is a (total) mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\overline{x}$ is referred to as the *contrary* of x.

We will assume that the inference rules in $\mathcal{R}$ have the syntax $l_0 \leftarrow l_1, \ldots l_n$ (for $n \geq 0$) where $l_i \in \mathcal{L}$. We will refer to l_0 and $l_1, \ldots l_n$ as the *head* and the *body* of the rule, respectively. We will represent the rule $l \leftarrow$ simply as l. As in [7], we will restrict attention to *flat* ABA frameworks, such that if $l \in \mathcal{A}$, then there exists no inference rule of the form $l \leftarrow l_1, \ldots, l_n \in \mathcal{R}$, for any $n \geq 0$.

We will adopt a generalisation of ABA frameworks, first given in [10], whereby assumptions allow *multiple contraries* (i.e. $\overline{\ }$ is a (total) mapping from $\mathcal{A}$ into $\wp(\mathcal{L})$). As argued in [10], multiple contraries are a useful generalisation to ease representation and comprehension of ABA frameworks. However, they do not really extend the expressive power of ABA frameworks. Indeed, there is a one-to-one correspondence between original ABA frameworks and ABA frameworks with multiple contraries. For example, consider the framework with multiple contraries $\langle \mathcal{L}_g, \mathcal{R}_g, \mathcal{A}_g, \overline{\ } \rangle$ where:

$\mathcal{L}_g = \{p, a, c_1, b, c_3, c_4\}$, $\mathcal{R}_g = \{p \leftarrow a; c_1 \leftarrow b; c_3\}$, $\mathcal{A}_g = \{a, b\}$,
$\overline{a} = \{c_1, c_2, c_3\}$ and $\overline{b} = \{c_4\}$

This can be turned into a corresponding original framework $\langle \mathcal{L}_o, \mathcal{R}_o, \mathcal{A}_o, \overline{\ } \rangle$ where:

$\mathcal{L}_o = \mathcal{L}_g \cup \{aux\}$, $\mathcal{R}_o = \mathcal{R}_g \cup \{aux \leftarrow c_1; aux \leftarrow c_2; aux \leftarrow c_3\}$, $\mathcal{A}_o = \mathcal{A}_g$,
$\overline{a} = aux$ and $\overline{b} = c_4$

We will use this correspondence to simplify some of the proofs in Section 4.

Given an ABA framework, an *argument* in favour of a sentence $x \in \mathcal{L}$ supported by a set of assumptions X, denoted $X \vdash x$, is a (backward) deduction from x to X, obtained by applying backwards the rules in $\mathcal{R}$. For example, given $\langle \mathcal{L}_g, \mathcal{R}_g, \mathcal{A}_g, \overline{\ } \rangle$ above, $\{a\} \vdash p$ is an argument.

In order to determine whether a conclusion (set of sentences) should be drawn, a set of assumptions needs to be identified providing an "acceptable" support for the conclusion. Various notions of "acceptable" support can be formalised, using a notion of "attack" amongst sets of assumptions whereby X_1 *attacks* X_2 iff there is an argument in favour of some $y \in \overline{x}$ supported by (a subset of) X_1 where x is in X_2 (for example, given $\langle \mathcal{L}_g, \mathcal{R}_g, \mathcal{A}_g, \overline{\ } \rangle$ above, $\{b\} \vdash c_1$ attacks $\{a\} \vdash p$). In this paper, we will consider the following notions of "acceptable" set of assumptions and support:

- a set of assumptions is *admissible* iff it does not attack itself and it attacks every set of assumptions attacking it;
- an *admissible support* for a claim is an admissible set X of assumptions such that there exists an argument in favour of the claim supported by a subset of X.

As shown in [9], there is a one-to-one correspondence between admissible supports for conclusions, in terms of sets of assumptions, and admissible sets of arguments (supported by assumptions), in the sense of [6].

AB-dispute derivations [7,9] allow to compute admissible supports for given claims (if any exists). They are finite sequences of tuples $\langle \mathcal{P}_i, \mathcal{O}_i, D_i, C_i \rangle$ where $\mathcal{P}_i$ and $\mathcal{O}_i$ represent (the set of sentences and the set of sets of sentences held by) the proponent and opponent (respectively) at step i in the dispute, D_i is the set of assumptions generated by the proponent in support of the initial claim and to defend itself against the opponent, and C_i is the set of assumptions in counter-arguments generated by the opponent that the proponent has chosen as "culprits" to counter-attack. The tuple at the start of a derivation for claim x is: $\langle \{x\}, \{\}, \mathcal{A} \cap \{x\}, \{\} \rangle$.

Example 1 *The AB-dispute derivation for the framework ($\mathcal{L}$ is omitted for brevity)*
$\mathcal{R} = \{p \leftarrow a, r; c_1 \leftarrow b, s; c_1 \leftarrow t; c_2 \leftarrow q; q; r \leftarrow e\}$, $\mathcal{A} = \{a, b, e\}$,
$\overline{a} = \{c_1\}$ *and* $\overline{b} = \{c_2\}$ *and* $\overline{e} = \{z\}$
computes an admissible support $\{a, e\}$ *for claim* p *as can be seen in table 1.*

AB-dispute derivations make use of a *selection function* to pick a sentence from the support of a "potential argument" to expand it further (if it is not an assumption) or to identify a possible point of attack (if it is an assumption). Such selection can be based on random choice, on whether or not the sentence is an assumption or on more complex criteria. The sentence selected by the chosen selection function at each step is underlined in the derivation in table 1. AB-dispute derivations implicitly compute a dialectical structure of ("potential") arguments. For the derivation in table 1, these arguments are (1)

$Step$	$Proponent$	$Opponent$	$DefenseSet$	$Culprits$
0	$\underline{\mathbf{p}}$			
1	$\underline{a}, r$		a	
2	r	$\{\underline{c1}\}$	a	
3	r	$\{\underline{b}, s\}, \{t\}$	a	
4	$\underline{c2}, r$	$\{t\}$	a	b
5	$\underline{q}, r$	$\{t\}$	a	b
6	$\underline{r}$	$\{t\}$	a	b
7	$\underline{e}$	$\{t\}$	a, e	b
8		$\{\underline{t}\}, \{z\}$	a, e	b
9		$\{\underline{z}\}$	a, e	b
10			**a, e**	b

Table 1. AB-dispute derivation for example 1.

$\{a, e\} \vdash p$, (2) $\{b, s\} \vdash c_1$, (3) $\{\} \vdash c_2$, where (3) attacks (2) and (2) "potentially" attacks (1) (here, (2) is a "potential" argument as its support contains a non-assumption s - see section 3 for a formal definition). This structure however is hidden in the computation. In [11], we presented structured AB-dispute derivations computing this structure in the case of *patient* selection functions, namely always selecting non-assumptions first (the selection function used in table 1 is not patient). Below, we define *generalised structured AB-dispute derivations* working with any selection function. We also formalise many of the concepts intuitively used in [11] and prove some formal results for the generalised structured AB-dispute derivations we define here.

3. Generalised structured AB-dispute derivations

Our *generalised structured AB-dispute derivations*, that we refer to simply as structured AB-dispute derivations, are sequences of tuples of the form $\langle \mathcal{P}_i, \mathcal{O}_i, D_i, C_i, A_i, R_i \rangle$. The elements D_i and C_i are the defense set and the set of culprits, exactly as in the original AB-dispute derivations. The elements $\mathcal{P}_i$ and $\mathcal{O}_i$, as before, represent the state of the proponent and opponent, but they are no longer sets of sentences and sets of sets of sentences, respectively. Instead, they consist of "potential arguments" together with information about which arguments they "potentially attack".

Definition 1 *A* potential argument $(Y, X \vdash x)$ *in favour of a sentence* $x \in \mathcal{L}$ *supported by* (Y, X)*, with* $X \subseteq \mathcal{A}$ *and* $Y \subseteq \mathcal{L}$ *is a (backward) deduction from* x *to* $X \cup Y$*, obtained by applying backwards the rules in* $\mathcal{R}$*.*

Trivially, a potential argument $(\{\}, X \vdash x)$ corresponds to an argument $X \vdash x$ as in conventional ABA. Below, we will refer to arguments in conventional ABA as *actual arguments*. A potential argument $(Y, X \vdash x)$ with $Y \subseteq \mathcal{A}$ also corresponds to an actual argument $X \cup Y \vdash x$.

Intuitively, potential arguments correspond to intermediate stages in the construction of actual arguments. It may be possible to turn a potential argument into zero, one, or many actual arguments, depending on whether all non-assumptions in Y can be reduced to assumptions via the backward application of rules in $\mathcal{R}$. In example 1, $(\{r\}, \{a\} \vdash p)$

is a potential argument that can be turned into a single actual argument $\{a, e\} \vdash p$ and $(\{s\}, \{b\} \vdash c_1)$ is a potential argument that cannot be turned into an actual argument.

The notion of attack between (actual) arguments can be generalised to a notion of "potential attack" between potential arguments:

Definition 2 *A potential argument* $(Y_1, X_1 \vdash x_1)$ potentially attacks *a potential argument* $(Y_2, X_2 \vdash x_2)$ *iff* $x_1 \in \overline{z}$ *for some* $z \in (Y_2 \cap \mathcal{A}) \cup X_2$.

Trivially, potential attacks correspond to actual attacks whenever the potential arguments correspond to actual arguments.

The two new elements in structured AB-dispute derivations, A_i and R_i, hold, respectively, the currently computed (potential and actual) arguments and a binary relation between these arguments, corresponding to (potential and actual) attacks. In order to ease the representation of R_i, we adopt a labelling convention for arguments, namely

- $\mathcal{P}_i$ and $\mathcal{O}_i$ are sets of expressions of the form: $id : (Y, X \vdash x) \rightsquigarrow id^*$ indicating a (potential or actual) argument $(Y, X \vdash x)$ labelled id (potentially or actually) attacking another argument labelled id^*
- A_i consists of expressions of the form $id : (Y, X \vdash x)$ representing a (potential or actual) argument $(Y, X \vdash x)$ labelled id
- R_i is a set of expressions of the form $id \rightsquigarrow id^*$ standing for "the (potential or actual) argument labelled by id (potentially or actually) attacks the (potential or actual) argument labelled by id^*"

Structured AB-dispute derivations interleave the construction of arguments and their evaluation (with respect to the admissibility semantics) and thus need to store potential arguments (in the components $\mathcal{P}_i$ and $\mathcal{O}_i$). Once these arguments are evaluated (with respect to admissibility) they are eliminated from $\mathcal{P}_i$ or $\mathcal{O}_i$ and stored in A_i (with R_i also appropriately modified).

For the purpose of labelling arguments, we will assume the existence of a function $newLabel()$ that returns a fresh label every time it is invoked. We will adopt a marking mechanism for specifying our structured AB-dispute derivations, so that sentences in the support of arguments in $\mathcal{P}_i$ and $\mathcal{O}_i$ are marked after they have been selected (by the selection function). We will assume that the selection function will always select a sentence in the unmarked bit of the support of a potential argument. This is similar to the marking mechanism adopted for elements of $\mathcal{O}_i$ in IB-dispute derivations in [9]. However, here we store marked assumptions in the X component of the support of a potential argument $(Y, X \vdash x)$, and unmarked elements in its Y component. Moreover, here the marking is needed in order to compute the dialectical structures, whereas in [9] it was needed to ensure the correctness of IB-dispute derivations.

Definition 3 *A* (generalised) structured AB-dispute derivation of a defense set D *and* of a dialectical structure (A, R) for a sentence $\alpha \in \mathcal{L}$ *is a finite sequence of* tuples $\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0, A_0, R_0 \rangle, \ldots, \langle \mathcal{P}_i, \mathcal{O}_i, D_i, C_i, A_i, R_i \rangle, \ldots, \langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n, A_n, R_n \rangle$ *where initially*

$\mathcal{P}_0 = \{l_1 : ([\{\alpha\}, \{\}] \vdash \alpha) \rightsquigarrow \emptyset\}$ *where* $l_1 = newLabel()$ *and* $\emptyset$ *is a special label representing that this does not attack any other arguments*

$D_0 = \mathcal{A} \cap \{\alpha\}$
$\mathcal{O}_0 = C_0 = A_0 = R_0 = \{\}$

and upon termination

$\mathcal{P}_n = \mathcal{O}_n = \{\}$
$D = D_n$, $A = A_n$, $R = R_n$

and for every $0 \leq i < n$, *one potential argument* $curArg$, *of the form* $l : ([S_u, S_m] \vdash G) \rightsquigarrow l_e$, *is chosen in either* $\mathcal{P}_i$ *or* $\mathcal{O}_i$, *a sentence* σ *is selected in* S_u, *and:*

1. *If* $curArg$ *is chosen in* $\mathcal{P}_i$ *then*

   (i) if σ *is an assumption then*

   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{curArg\} \cup newP$
   $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{new : ([\{x\}, \{\}] \vdash x) \rightsquigarrow l \mid x \in \overline{\sigma}$ *and new = newLabel()*$\}$
   $D_{i+1} = D_i$
   $C_{i+1} = C_i$
   $A_{i+1} = A_i \cup newArg$
   $R_{i+1} = R_i \cup newRel$

   where $newP, newArg, newRel$ *are defined in the table below, separating out whether or not the selected* σ *was the last unmarked element in the premise of* $curArg$ *(i.e.* $S_u = \{\sigma\}$*)*

$S_u - \{\sigma\} = \{\}$	$S_u - \{\sigma\} \neq \{\}$
$newP = \{\}$ $newArg = \{l : ([\{\}, S_m \cup \{\sigma\}] \vdash G)\}$ $newRel = \{l \rightsquigarrow l_e\}$	$newP = \{l : ([S_u - \{\sigma\}, S_m \cup \{\sigma\}] \vdash G) \rightsquigarrow l_e\}$ $newArg = \{\}$ $newRel = \{\}$

   (ii) if σ *is a non-assumption and there exists some rule of the form* $\sigma \leftarrow B \in \mathcal{R}$ *such that* $C_i \cap B = \{\}$ *then* [1]

   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{curArg\} \cup newP$
   $\mathcal{O}_{i+1} = \mathcal{O}_i$
   $D_{i+1} = D_i \cup (\mathcal{A} \cap B)$
   $C_{i+1} = C_i$
   $A_{i+1} = A_i \cup newArg$
   $R_{i+1} = R_i \cup newRel$

   where $newP, newArg, newRel$ *are defined in the table below*

$(S_u - \{\sigma\}) \cup (B - D_i) = \{\}$	$(S_u - \{\sigma\}) \cup (B - D_i) \neq \{\}$
$newP = \{\}$ $newArg = \{l : ([\{\}, S_m \cup B] \vdash G)\}$ $newRel = \{l \rightsquigarrow l_e\}$	$newP = \{l : ([S'_u, S'_m] \vdash G) \rightsquigarrow l_e\}$ $newArg = \{\}$ $newRel = \{\}$ *where* $S'_u = (S_u - \{\sigma\}) \cup (B - D_i)$ *and* $S'_m = S_m \cup (B \cap D_i)$

[1] We treat B and all bodies of inference rules in $\mathcal{R}$ as sets.

2. *If $curArg$ is chosen from $\mathcal{O}_i$ then*

(i) if σ is an assumption then

(a) either σ is ignored, i.e.

$\mathcal{P}_{i+1} = \mathcal{P}_i$
$\mathcal{O}_{i+1} = \mathcal{O}_i - \{curArg\} \cup \{l : ([(S_u - \{\sigma\}), (S_m \cup \{\sigma\})] \vdash G) \rightsquigarrow l_e\}$
$D_{i+1} = D_i$
$C_{i+1} = C_i$
$A_{i+1} = A_i$
$R_{i+1} = R_i$

(b) or $\sigma \in C_i$ and $\sigma \notin D_i$

$\mathcal{P}_{i+1} = \mathcal{P}_i$
$\mathcal{O}_{i+1} = \mathcal{O}_i - \{curArg\}$
$D_{i+1} = D_i$
$C_{i+1} = C_i$
$A_{i+1} = A_i \cup \{l : ([S_u - \{\sigma\}, S_m \cup \{\sigma\}] \vdash G)\}$
$R_{i+1} = R_i \cup \{l \rightsquigarrow l_e\} \cup \{someLabel \rightsquigarrow l\}$

where $someLabel$ is any label such that, for some value of X, Y, Z and for some $x \in \overline{\sigma}$, either $someLabel : ([X, Y] \vdash x) \in A_i$ or $someLabel : ([\{X\}, \{Y\}] \vdash x) \rightsquigarrow Z \in P_i$. [2]

(c) or $\sigma \notin C_i$ and $\sigma \notin D_i$ and

$\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{new : ([\{x\}, \{\}] \vdash x) \rightsquigarrow l\}$
where $x \in \overline{\sigma}$ and new = newLabel()
$\mathcal{O}_{i+1} = \mathcal{O}_i - \{curArg\}$
$D_{i+1} = D_i \cup (\mathcal{A} \cap \{x\})$
$C_{i+1} = C_i \cup \{\sigma\}$
$A_{i+1} = A_i \cup \{l : ([S_u - \{\sigma\}, S_m \cup \{\sigma\}] \vdash G)\}$
$R_{i+1} = R_i \cup \{l \rightsquigarrow l_e\}$

(ii) if σ is a non-assumption, then

$\mathcal{P}_{i+1} = \mathcal{P}_i$
$\mathcal{O}_{i+1} = \mathcal{O}_i - \{curArg\} \cup \{l : ([(S_u - \{\sigma\} \cup B), S_m] \vdash G) \rightsquigarrow l_e \mid \sigma \leftarrow B \in \mathcal{R}$ *and* $B \cap C_i = \{\}\ \}$
$D_{i+1} = D_i$
$C_{i+1} = C_i$
$A_{i+1} = A_i \cup \{n : ([(S_u - \{\sigma\}) \cup (B - C_i), S_m \cup (B \cap C_i)] \vdash G) \mid n = newLabel()$ *and* $\sigma \leftarrow B \in \mathcal{R}$ *and* $B \cap C_i \neq \{\}\ \}$
$R_{i+1} = R_i \cup \{m \rightsquigarrow n \mid \sigma \leftarrow B \in \mathcal{R}$ *and* $B \cap C_i \neq \{\}$ *and* $m =$ ***find_label***$(B \cap C_i)\}$
$\cup \{n \rightsquigarrow l_e \mid n : ([S'_u, S'_m] \vdash l) \in A_{i+1} - A_i\}$

where ***find_label***$(Set) = someLbl$ *such that* $\omega \in Set$ *and*
$((someLbl : ([X, Y] \vdash \omega)) \in A_i$ *or* $(someLbl : ([X, Y] \vdash \omega) \rightsquigarrow Z) \in P_i)$

[2] If $\sigma \in C_i$, either the culprit σ is already defeated or it is on the proponent's agenda of things to be defeated. One must search through both P_i and A_i to find $someLabel$.

Intuitively, three choices have to be made at each step in a derivation. First a (fictional) $player$ must be chosen: either the proponent ($\mathcal{P}_i$) or the opponent ($\mathcal{O}_i$). Next, from the chosen set, one (potential or actual) argument $curArg$ needs to be chosen for further consideration. $curArg$ will be of the form $l : ([S_u, S_m] \vdash G) \rightsquigarrow l_e$. Finally, one element σ from (the unmarked part S_u of) the support of $curArg$ is selected. There are now four main cases to consider, depending on the player and whether σ is an assumption or not.

In case 1(i), the proponent plays and σ is an assumption: this is simply marked, and new potential arguments for the contraries of σ are added to the opponent. Moreover, if σ is the last unmarked element, the dialectical structure also gets augmented (note that in this case the argument added to A_i is an actual argument).

In case 1(ii), the proponent plays and σ is a non-assumption: this is unfolded using a rule with body B. If B is empty or all elements in B are assumptions that have already been defended (i.e. they are in D_i), then the dialectical structure gets augmented (note that in this case the argument added to A_i is an actual argument). Otherwise, σ in $curArg$ is unfolded to the rule body B: The part of B that is already in the defense set is added to the marked elements S_m (and hence treated as if already considered), whereas the part of B that is not yet in the defense set is added to S_u for future consideration.

In case 2(i), the opponent plays and σ is an assumption. If $\sigma \in D_i$, then the only option is to ignore it (case 2ia), as choosing such a σ as a culprit would make the defense set attack itself and hence not be admissible.

If however $\sigma \notin D_i$, then it could be a culprit (but note that it could also be ignored). If σ is already a known culprit ($\sigma \in C_i$), then case 2ib applies and the potential attack $curArg$ can be moved to A_i (and R_i be appropriately modified, too) since either σ has already been defeated or the fact that it needs to be defeated must already have been "stored" in $\mathcal{P}_i$. Here, the argument defeating σ is labelled with $someLabel$.

If σ is not a known culprit yet ($\sigma \notin C_i$), then we add it to the set of culprits and pick one of its contraries (say, x) for the proponent to show (in order to thereby counter-attack $curArg$). Note that, for a derivation to exist, all potential arguments in all $\mathcal{O}_i$ need to be defeated, as otherwise the termination condition $\mathcal{O}_n = \{\}$ would not be met. If some chosen culprit cannot be defeated, then the implementation of the algorithm can resort to backtracking on some of the choices (either the culprit itself, that can be ignored, or the contrary of the chosen culprit, or one of the rules at step 1(ii) for generating the argument attacking the culprit).

In case 2(ii), the opponent plays and σ is a non-assumption. Here, the opponent expands σ in all possible ways (i.e. using all possible rules). $curArg$ is replaced with many new potential arguments, one for each applicable rule which has no known culprit in its body. For each applicable rule which has some known culprit in its body, we extend the dialectical structure by adding to A_i one potential argument for each rule that had culprits in its body. R_i is also augmented appropriately.

Let us now consider the ABA framework in example 1. A structured AB-dispute derivation exists for claim p, e.g. as given table 2 (others exist for other choices of players and selection functions, here we have used the same choices and selection functions as in table 1). Note that we obtain the same number of steps, the same defense set and the same set of culprits as for ordinary AB-dispute derivations (table 1).

#	*Proponent*	*Opponent*	D	C	A	R
0	$l_1 : ([\{p\}, \{\}] \vdash p) \rightsquigarrow \emptyset$					
1	$l_1 : ([\{a, r\}, \{\}] \vdash p) \rightsquigarrow \emptyset$		a			
2	$l_1 : ([\{r\}, \{a\}] \vdash p) \rightsquigarrow \emptyset$	$l_2 : ([\{c1\}, \{\}] \vdash c1) \rightsquigarrow l_1$	a			
3	$l_1 : ([\{r\}, \{a\}] \vdash p) \rightsquigarrow \emptyset$	$l_{2.1} : ([\{b, s\}, \{\}] \vdash c1) \rightsquigarrow l_1$, $l_{2.2} : ([\{t\}, \{\}] \vdash c1) \rightsquigarrow l_1$	a			
4	$l_3 : ([\{c2\}, \{\}] \vdash c2) \rightsquigarrow l_{2.1}$, $l_1 : ([\{r\}, \{a\}] \vdash p) \rightsquigarrow \emptyset$	$l_{2.2} : ([\{t\}, \{\}] \vdash c1) \rightsquigarrow l_1$	a	b	$l_{2.1} : ([\{s\}, \{b\}] \vdash c_1)$	$l_{2.1} \rightsquigarrow l_1$
5	$l_3 : ([\{q\}, \{\}] \vdash c2) \rightsquigarrow l_{2.1}$, $l_1 : ([\{r\}, \{a\}] \vdash p) \rightsquigarrow \emptyset$	$l_{2.2} : ([\{t\}, \{\}] \vdash c1) \rightsquigarrow l_1$	a	b	$l_{2.1} : ([\{s\}, \{b\}] \vdash c_1)$	$l_{2.1} \rightsquigarrow l_1$
6	$l_1 : ([\{r\}, \{a\}] \vdash p) \rightsquigarrow \emptyset$	$l_{2.2} : ([\{t\}, \{\}] \vdash c1) \rightsquigarrow l_1$	a	b	$l_{2.1} : ([\{s\}, \{b\}] \vdash c_1)$, $l_3 : ([\{\}, \{\}] \vdash c_2)$	$l_{2.1} \rightsquigarrow l_1$, $l_3 \rightsquigarrow l_{2.1}$
7	$l_1 : ([\{e\}, \{a\}] \vdash p) \rightsquigarrow \emptyset$	$l_{2.2} : ([\{t\}, \{\}] \vdash c1) \rightsquigarrow l_1$	a, e	b	$l_{2.1} : ([\{s\}, \{b\}] \vdash c_1)$, $l_3 : ([\{\}, \{\}] \vdash c_2)$	$l_{2.1} \rightsquigarrow l_1$, $l_3 \rightsquigarrow l_{2.1}$
8		$l_{2.2} : ([\{t\}, \{\}] \vdash c1) \rightsquigarrow l_1$, $l_4 : ([\{z\}, \{\}] \vdash z) \rightsquigarrow l_1$	a, e	b	$l_{2.1} : ([\{s\}, \{b\}] \vdash c_1)$, $l_3 : ([\{\}, \{\}] \vdash c_2)$, $l_1 : ([\{\}, \{a, e\}] \vdash p)$	$l_{2.1} \rightsquigarrow l_1$, $l_3 \rightsquigarrow l_{2.1}$, $l_1 \rightsquigarrow \emptyset$
9		$l_4 : ([\{z\}, \{\}] \vdash z) \rightsquigarrow l_1$	a, e	b	$l_{2.1} : ([\{s\}, \{b\}] \vdash c_1)$, $l_3 : ([\{\}, \{\}] \vdash c_2)$, $l_1 : ([\{\}, \{a, e\}] \vdash p)$	$l_{2.1} \rightsquigarrow l_1$, $l_3 \rightsquigarrow l_{2.1}$, $l_1 \rightsquigarrow \emptyset$
10			a, e	b	$l_{2.1} : ([\{s\}, \{b\}] \vdash c_1)$, $l_3 : ([\{\}, \{\}] \vdash c_2)$, $l_1 : ([\{\}, \{a, e\}] \vdash p)$	$l_{2.1} \rightsquigarrow l_1$, $l_3 \rightsquigarrow l_{2.1}$, $l_1 \rightsquigarrow \emptyset$

Table 2. Structured AB-dispute derivation for example 1.

The following realistic example illustrate the possible use of structured AB-dispute derivations in real-world setting. Imagine a scenarios whereby parents are trying to find a name for their new-born baby. Let us assume that parents deem a name as acceptable if they both like it and it is easy to remember. Also, Dad dislikes common names and he also does not want the baby to have the same name as his uncle. Mom on the other hand by default hates all names unless she explicitly approves of them. This scenario can be expressed as the following ABA framework [3]:

$\mathcal{L} = \{p(t) \mid p \in Preds \ \ and \ t \in Terms\}$ where $Preds$ are all predicates occurring in $\mathcal{R}$ and $Terms = \{adrian, vercingetorix\}$

$\mathcal{A} = \{all\_like(Name), mom\_hates(Name) | Name \in Terms\}$

$$\mathcal{R} = \left\{ \begin{array}{l} acceptable(Name) \leftarrow all\_like(Name), easy\_to\_remember(Name); \\ easy\_to\_remember(Name) \leftarrow short(Name); \\ some\_dislike(Name) \leftarrow mom\_hates(Name); \\ some\_dislike(Name) \leftarrow dad\_hates(Name); \\ dad\_hates(Name) \leftarrow too\_commom(Name); \\ dad\_hates(Name) \leftarrow uncle\_has(Name); \\ mom\_not\_hate(Name) \leftarrow mom\_said\_ok(Name); \\ mom\_said\_ok(adrian); \\ short(adrian) \end{array} \right\}$$

[3]The rules containing variables, i.e. words beginning with an upper case letter, are short-hand for all their ground instances (e.g. all rules where *Name* is instantiated to *adrian*).

There exists a structured AB-dispute derivation of defence set D_{11} and of dialelectical structure (A_{11}, D_{11}) for $acceptable(adrian)$ where

$D_{11} = \{all\_like(adrian)\}$ and $R_{11} = \{l_1 \rightsquigarrow \emptyset, l_2 \rightsquigarrow l_1, l_3 \rightsquigarrow l_2\}$ and
$A_{11} = \{l_1 : ([\{\}, \{all\_like(adrian)\}] \vdash acceptable(adrian)),$
$l_2 : ([\{\}, \{mom\_hates(adrian)\}] \vdash some\_dislike(adrian)),$
$l_3 : ([\{\}, \{\}] \vdash mom\_not\_hate(adrian))\}$

Note that all computed arguments are actual arguments. The dialectical structure can be graphically represented as follows (here $\leftarrow$ stands for attack):

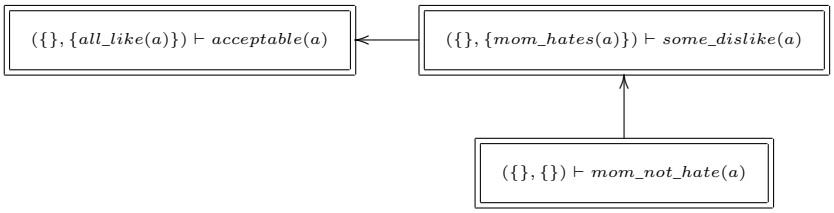

Figure 1. Dialectical structures for the realistic example.

4. Results

In this section, we state and sketch formal proofs of the following main results [4]: 1) our structured AB-dispute derivations are a direct generalisation of the AB-dispute derivations of (a variant of) [7,8,9] (see theorem 1 below); 2) structured AB-dispute derivations compute admissible supports of the input claims (see corollary 1 below); 3) the dialectical structure computed by structured AB-dispute derivations can be mapped onto admissible dispute trees [7,9] (see theorem 2 below). We prove the results for ABA frameworks where the contrary of every assumption is a singleton set (as in original ABA). As discussed in section 2, this can be done without loss of generality.

Theorem 1 *For any structured AB-dispute derivation of a defense set D and of dialectical structure (A, R) for a claim α there exists an AB-dispute derivation of a defense set D for α.*

Note that the converse also holds, namely existence of an AB-dispute derivation guarantees existence of a structured AB-dispute derivation computing the same defense set (and some dialectical structure).

The proof of theorem 1 uses a variant of AB-dispute derivation, equivalent to the one in [9]. This variant combines two cases in AB-dispute derivations in [9], cases 2ic.1 and 2ic.2, into one single case 2ic, corresponding to case 2ic in definition 3. This variant is equivalent to the original version under the restriction that, when the contrary of an assumption is an assumption, its original contrary is the original assumption. This restriction also held (implicitly) in [9]. The proof of theorem 1 is constructive, in that it uses mappings from the $\mathcal{P}_i$ and $\mathcal{O}_i$ components in structured AB-dispute derivations onto $\mathcal{P}_i$ and $\mathcal{O}_i$ components in AB-dispute derivations: these mappings turn sets of po-

[4] Full proofs can be found in an accompanying technical report.

tential arguments into sets of assumptions (for $\mathcal{P}_i$; these are all the assumptions in the unmarked part of the support of the potential arguments) and sets of sets of assumptions (for $\mathcal{O}_i$; each such set is the unmarked part of the support of one potential argument). No mappings are required for the D_i and C_i components - that are identical in the two kinds of derivations, or the A_i and R_i components, that are absent in the original AB-dispute derivations.

Corollary 1 *Given any structured AB-dispute derivation of a defense set D and of a dialectical structure (A, R) for a sentence α, D is an admissible support for α.*

This result follows directly from theorem 1 and the correctness of AB-dispute derivations in [7,9].

Theorem 2 below sanctions that the dialectical structure computed by structured AB-dispute derivations can be mapped onto admissible dispute trees [7,9], defined as follows.

Definition 4 *(definitions 3.1 and 3.2 in [9]) A* dispute tree *for an argument a is a (possibly infinite) tree $\mathcal{T}$ such that*

1. *Every node of $\mathcal{T}$ is labelled by an argument and is assigned the status of* proponent *node or* opponent *node, but not both.*
2. *The root is a proponent node labelled by a.*
3. *For every proponent node N labelled by an argument b, and for every argument c that attacks b, there exists a child of N, which is an opponent node labelled by c.*
4. *For every opponent node N labelled by an argument b, there exists exactly one child of N which is a proponent node labelled by an argument which attacks b.*
5. *There are no other nodes in $\mathcal{T}$ except those given by 1-4 above.*

The set of all assumptions belonging to the proponent nodes in $\mathcal{T}$ is called the defense set *of $\mathcal{T}$. A dispute tree is* admissible *iff no argument labels both a proponent and an opponent node.*

Given a dialectical structure (A, R) computed by a structured AB-dispute derivation, let $Actual(A, R)$ stand for the pair (A^*, R^*) consisting of the set A^* of all actual arguments that can be obtained from A (by backward deduction from the potential arguments in A) and R^* the restriction of R to elements of A^*. Moreover, given a dialectical structure $(Args, Rel)$, $\mathcal{T}(Args, Rel)$ will refer to the tree constructed as follows: the root is the argument attacking $\emptyset$ in $Args$; nodes are in correspondence with elements of $Args$; x is a child of y iff $(x, y) \in Rel$. Then,

Theorem 2 *For any structured AB-dispute derivation of a defense set D and of dialectical structure (A, R) for a claim α, let $Actual(A, R) = (A^*, R^*)$ and $\mathcal{T} = \mathcal{T}(A^*, R^*)$. Then, $\mathcal{T}$ is an admissible dispute tree for α with defense set D' such that $D' \subseteq D$.*

The proof of this theorem relies upon a number of lemmas, including:

Lemma 1 *For each structured AB-dispute derivation of a defense set D and of dialectical structure (A, R) for a claim α, let C be the final set of culprits. For every $x \in C$ there exists an argument in A attacking x.*

Lemma 2 *For each structured AB-dispute derivation of a defense set D and of dialectical structure (A, R) for a claim α, all the actual arguments that one can build from any potential argument in A are attacked by some argument in A.*

Lemma 3 *For each structured AB-dispute derivation of a defense set D and of dialectical structure (A, R) for a claim α, let C be the final set of culprits. Every potential argument in A of the form $(S_u, S_m \vdash x)$ such that $S_u \neq \{\}$ has the property that $(X_u \cup X_m) \cap C \neq \{\}$.*

5. Conclusions

We have presented a computational model for a form of argumentation that is a hybrid between abstract and assumption-based argumentation. To the best of our knowledge, this work is the first attempt to combine the two forms of argumentation in a synergetic and practical manner, building upon [7,9,11]. Our hybrid computational model would be beneficial to developing applications of argumentation where explicit justifications of claims in terms of full dialectical structures are required, for example, for the purpose of argumentation-based negotiation, as it would provide the negotiating agents with an explicit structure of the dialectical process.

Computational models have been proposed for abstract argumentation, such as the Two Party Immediate response (TPI) disputes [14] and the dialectical proof theories of [4,13]. Like ours, these models can be seen as games between two fictional players in which the proponent always acts first. These models compute different semantics than admissibility (i.e. preferred [4,14], robust and defensible [13] semantics) for a different argumentation framework (i.e. abstract argumentation). In particular, these computational models do not need to construct arguments, as arguments are seen as black boxes within abstract argumentation. Moreover, although they use a dialectical protocol similar to the one underlying the generalised AB-dispute derivation, [4,13] insist on and [14] implies strictly alternating turns between proponent and opponent, whereas we do not do so in order to allow interleaving the construction of arguments and the analysis of their dialectical status.

Compared to existing computational models for assumption-based argumentation (e.g. [9]), our computational model manipulates potential, rather than actual, arguments. These correspond to stages in the construction of actual arguments, and allow the interleaving of the construction of (actual) arguments and their evaluation (with respect to the admissibility semantics). As a result, the set of arguments returned by our computational model may include potential arguments, that have been defeated before being fully constructed. This may seem as a departure from conventional abstract argumentation. Note however that the computational model relies upon a selection function for deciding how arguments are constructed, using backward deductions. As noted in [7], when this selection function is *patient*, the computed arguments are all guaranteed to be actual. In other words, our computational model generalises conventional abstract argumentation without changing its spirit.

Structured AB-dispute derivations have been implemented in the CaSAPI (Credulous and Sceptical Argumentation: Prolog Implementation) system (in its current version v4.3) which can be downloaded from **www.doc.ic.ac.uk/∼dg00/casapi.html**. Previous

versions were inspired by traditional assumption-based argumentation (version 2, [10]) and by a restricted form of hybrid argumentation (version 3, [11]).

In the near future we aim at improving the practical aspects of hybrid argumentation, by extensive experimentation with the CaSAPI system and by extending its graphical user interface. We also plan to compare this system with a number of other argumentation systems, including Gorgias [5], for credulous argumentation in argumentation frameworks with preferences amongst defeasible rules, the ASPIC system [3] dealing with quantitative uncertainty, DeLP [12] for defeasible logic programming, and the system by Krause et al. [2]. A mapping from these various frameworks onto assumption-based argumentation (possibly extended) is needed in order to carry out a full comparison.

Finally, we also plan to extend (theoretically and practically) the computational machinery in this paper to compute sceptical semantics for argumentation, namely the grounded and ideal semantics, already implemented in version 2 of CaSAPI.

Acknowledgements

This work was partially funded by the Sixth Framework IST programme of the EC, under the 035200 ARGUGRID project.

References

[1] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1-2):63–101, 1997.

[2] D. Bryant and P. Krause. An implementation of a lightweight argumentation engine for agent applications. In *Proc. of JELIA06*, 2006.

[3] M. Caminada, S. Doutre, S. Modgil, H. Prakken, and G.A.W. Vreeswijk. Implementations of argument-based inference. In *Rev. of Argumentation Tech.*, 2004.

[4] Claudette Cayrol, Sylvie Doutre, and Jérôme Mengin. On Decision Problems Related to the Preferred Semantics for Argumentation Frameworks. *Journal of Logic and Computation*, 13(3):377–403, 2003.

[5] N. Demetriou and A. C. Kakas. Argumentation with abduction. In *Proceedings of the fourth Panhellenic Symposium on Logic*, 2003.

[6] P.M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[7] P.M. Dung, R.A. Kowalski, and F. Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Art. Int.*, 170:114–159, 2006.

[8] P.M. Dung, P. Mancarella, and F. Toni. A dialectic procedure for sceptical, assumption-based argumentation. In *Proc, of the 1st Int'l Conference on Computational Models of Argument*, 2006.

[9] P.M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence - Special Issue on Argumentation in Artificial Intelligence*, 171(10-15):642–674, July-October 2007.

[10] D. Gaertner and F. Toni. CaSAPI: a system for credulous and sceptical argumentation. In *Proc. LPNMR Workshop on Argumentation for Non-monotonic Reasoning*, pages 80–95, 2007.

[11] D. Gaertner and F. Toni. Computing arguments and attacks in assumption-based argumentation. *IEEE Intelligent Systems*, 22(6), 2007.

[12] A. Garcia and G. Simari. Defeasible logic programming: An argumentative approach. *Journal of Theory and Practice of Logic Prog.*, 4(1-2):95–138, 2004.

[13] Hadassa Jakobovits and Dirk Vermeir. Dialectic semantics for argumentation frameworks. In *International Conference on Artificial Intelligence and Law*, pages 53–62, 1999.

[14] G. Vreeswijk and H. Prakken. Credulous and sceptical argument games for preferred semantics. In *Proc. JELIA*, volume 1919 of *LNCS*, pages 224–238. Springer Verlag, 2000.