Assumption-Based Argumentation for Communicating Agents

Adil Hussain and **Francesca Toni**
Department of Computing, Imperial College London
South Kensington Campus, London SW7 2AZ, UK
{ah02, ft}@imperial.ac.uk

Abstract

Assumption-Based Argumentation (ABA), and to a large extent argumentation in general, up to now has been considered in a single-agent setting. ABA, in particular, is such that an agent engages in a dispute (dialectic proof procedure) with itself (an imaginary opponent) to decide whether a claim is acceptable according to some acceptability criteria. We present in this paper a generalised proof procedure for the admissibility semantics of ABA, which is still a dispute by an agent with itself but such that the outcome can be readily communicated to other agents. This is important for applications in multi-agent systems wherein agents may differ in the knowledge they have and may need to communicate their arguments between one another to convince each other of the acceptability or not of a given claim.

1. Introduction

Assumption-based argumentation (ABA) (Dung, Kowalski, and Toni 2009) is a general-purpose framework for argumentation, where arguments 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 (dispute derivations) for ABA, e.g. (Dung, Mancarella, and Toni 2007), 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 extension). The dispute derivations find a set of assumptions, to defend a given claim, by starting from an initial set of assumptions that supports an argument for the claim and adding defending assumptions incrementally to counter-attack all attacks.

It is important to note that the dispute derivations of (Dung, Mancarella, and Toni 2007) take place within the mind of a single agent, between fictional proponent and opponent. Also, the resulting arguments (sets of assumptions) are not built to be communicated to other agents, but rather for an agent to determine within itself whether a claim is acceptable. In this paper, we focus on generalising the computational model for admissibility, called *AB-dispute derivations* (Dung, Mancarella, and Toni 2007). The computational model we present is still a dispute derivation within the mind of a single agent but such that the resulting arguments can be readily communicated, as enthymemes (Hunter 2007), to other agents. The benefits of allowing agents to exchange arguments can be found in many different types of dialogues, e.g. inquiry (Black and Hunter 2008), negotiation (Rahwan et al. 2003; Amgoud, Parsons, and Maudet 2000; Hussain and Toni 2008), deliberation (Mcburney, Hitchcock, and Parsons 2002) as well as of course persuasion (Amgoud, Maudet, and Parsons 2000; Prakken 2005). Thus a computational model that can generate communicable, acceptable arguments is clearly needed.

The paper is structured as follows: Section 2 presents background on ABA and admissibility. Section 3 presents our generalised ABA framework. Section 4 introduces an example that will be referred to throughout the paper. Sections 5 and 6 describe our generalised dispute derivation procedure used to build communicable arguments for a claim. Section 7 demonstrates how exchanging communicable arguments can be useful in dialogue between agents. Section 8 gives theoretical results. Lastly, Section 9 concludes.

2. Background

An ABA framework as defined in (Dung, Kowalski, and Toni 2009) is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, c \rangle$ 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*,
- c is a (total) mapping from $\mathcal{A}$ into $\mathcal{L}$, where $c(x)$ is referred to as the *contrary* of x.

We adopt a generalisation of ABA frameworks in this paper, as in (Gaertner and Toni 2008), whereby assumptions allow *multiple contraries* (i.e. c is a (total) mapping from $\mathcal{A}$ into $\wp(\mathcal{L}) - \{\emptyset\}$, where any $y \in c(x)$ is referred to as *a* contrary of x). We assume that the inference rules in $\mathcal{R}$ have the syntax $l_0 \leftarrow l_1, \dots l_n$ (for $n \geq 0$) where $l_i \in \mathcal{L}$. We represent $l_0 \leftarrow$ simply as l_0, and refer to this as a *fact*. As in (Dung, Mancarella, and Toni 2007) we restrict attention to *flat* ABA frameworks, such that if $l_0 \in \mathcal{A}$, then there exists no inference rule of the form $l_0 \leftarrow l_1, \ldots, l_n \in \mathcal{R}$, for any $n \geq 0$.

Our focus in this paper is the admissibility semantics and its computation. Firstly, an *argument* in favour of a sentence $p \in \mathcal{L}$ supported by $A \subseteq \mathcal{A}$, denoted $A \vdash p$, is a (backward) deduction from p to A, obtained by applying backwards the rules in $\mathcal{R}$. In (Dung, Mancarella, and Toni 2007) the notion of admissibility is formalised using a notion of "attack" among sets of assumptions whereby $X_1 \subseteq \mathcal{A}$ *attacks* $X_2 \subseteq \mathcal{A}$ iff, for some x is in X_2, there is an argument in favour of some $y \in c(x)$ supported by (a subset of) X_1. *A set of assumptions is admissible* iff it can counter-attack every attack (and does not attack itself). *A claim is admissible* then if it is the conclusion of an argument that is supported by a set of assumptions that can be extended to an admissible set of assumptions. The AB-dispute derivations of (Dung, Mancarella, and Toni 2007) prove that a claim is admissible by building an admissible set of assumptions.

3. Agent ABA Framework

In (Hunter 2007), it is argued that arguments presented by real-world agents normally only explicitly represent some of the premises for entailing their claim and/or they do not explicitly state their claim. It is argued that this is because there is some common knowledge that can be assumed by a proponent of an argument and the recipient of it. This allows the proponent of an argument to encode it by ignoring the common knowledge, and it allows a recipient of an argument to decode it by drawing on the common knowledge.

In the spirit of (Hunter 2007), with the view of allowing agents to exchange arguments and assuming that all agents in an *agent system* use the same common knowledge, we partition $\mathcal{R}$ into $\mathcal{R}_c$ and $\mathcal{R}^p$ (i.e. $\mathcal{R}_c \cap \mathcal{R}^p = \emptyset$, $\mathcal{R}_c \cup \mathcal{R}^p = \mathcal{R}$) and propose an *agent assumption-based argumentation* (AABA) framework $\langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}^p, \mathcal{A}, c\rangle$, where $\mathcal{R}_c$ is meant to contain knowledge common to all the agents and $\mathcal{R}^p$ is meant to contain personal knowledge of an agent not necessarily shared by other agents.

Definition 1 *An* AABA framework $\langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}^p, \mathcal{A}, c\rangle$ *is such that* $\mathcal{R}_c \cap \mathcal{R}^p = \emptyset$*, and* $\langle \mathcal{L}, \mathcal{R}_c \cup \mathcal{R}^p, \mathcal{A}, c\rangle$ *is an* ABA framework.

In what follows, we use $\mathcal{AF}$ to refer to an ABA framework $\langle \mathcal{L}, \mathcal{R}_c \cup \mathcal{R}^p, \mathcal{A}, c\rangle$ and $\mathcal{AF}_{Agent}$ to refer to an AABA framework $\langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}^p, \mathcal{A}, c\rangle$ for some $Agent$. We restrict $\mathcal{R}^p$ to containing facts only and refer to these as *personal facts*. Extending $\mathcal{R}^p$ to include inference rules of the form $l_0 \leftarrow l_1, \ldots, l_n$ (for $n > 0$) will be the subject of future work. Note that $\mathcal{A} \cap \mathcal{R}^p = \emptyset$, since we work with flat frameworks.

The generalised procedures for building *arguments*, which we present in Sections 5 and 6, are such that the necessary personal facts of an agent are accumulated, as well as the necessary assumptions. In standard ABA instead, only the assumptions are accumulated as arguments are constructed for internal consumption rather than to be communicated to other agents.

Note that we assume $\mathcal{L}$ (as well as $\mathcal{A}$ and c) is common to all agents in an agent system as communication (or, rather, *arguments*) exchanged between the agents in the system would consist of sentences in $\mathcal{L}$.

4. Example

We introduce in this section an example to be used throughout this paper to demonstrate our AABA framework. The example represents situations where agents may need to exchange appointments, e.g. on behalf of patients, and providing arguments would be useful for coming to agreement on whether to exchange an appointment or not.

Consider two AABA frameworks $\mathcal{AF}_x = \langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}^p_x, \mathcal{A}, c\rangle$ and $\mathcal{AF}_y = \langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}^p_y, \mathcal{A}, c\rangle$ for agents x and y respectively.

The language $\mathcal{L} \setminus \mathcal{A}$ consists of all ground instances of the following schemata:

- $sa(Ag1, Ag2, App1, App2)$, i.e. agents $Ag1$ and $Ag2$ should *swap appointments* $App1$ and $App2$;
- $r(Ag, G)$, i.e. agent Ag *requires* G to be fulfilled;
- $fr(App, G)$, i.e. appointment App *fulfils requirement* G;
- $l(Ag, App)$, i.e. agent Ag *likes* appointment App;
- $h(Ag, App)$, i.e. agent Ag *has* appointment App;
- $\neg cs(Ag, App1, App2)$, i.e. agent Ag *cannot swap* appointment $App1$ for $App2$.

The set of assumptions $\mathcal{A}$ consists of all ground instances of the following schemata:

- $\neg l(Ag, App)$, i.e. agent Ag *does not like* appointment App;
- $cs(Ag, App1, App2)$, i.e. agent Ag *can swap* appointment $App1$ for $App2$.

The contraries are as follows:

- $c(\neg l(Ag, App)) = \{l(Ag, App)\}$;
- $c(cs(Ag, App1, App2)) = \{\neg cs(Ag, App1, App2)\}$.

In general, agents have incomplete information about their environment, other agents, and so on. In this example, an agent may have incomplete information about what appointments other agents have or what they require, as well as which appointments fulfil which requirements. Assumptions allow an agent to reason under the uncertainty due to incomplete information. In particular, an agent can make assumptions about an agent (itself or another) not liking a certain appointment or being able to swap appointments until/unless it is told or knows otherwise.

$\mathcal{R}_c$ consists of inference rules *R1–R3* below. There, Ag, $Ag1$ and $Ag2$ can be any agents.

R1: $sa(Ag1, Ag2, App1, App2) \leftarrow h(Ag1, App1),$
$l(Ag1, App2),\ \neg l(Ag1, App1),\ cs(Ag2, App2, App1)$

Namely, $Ag1$ and $Ag2$ should swap $App1$ and $App2$ if, it is believed, $Ag1$ has $App1$ and $Ag1$ likes $App2$, and, it is assumed, $Ag1$ dislikes $App1$ and $Ag2$ can swap $App2$ for $App1$.

R2: $l(Ag, App) \leftarrow r(Ag, G),\ fr(App, G)$

Namely, Ag likes App if, it is believed, Ag requires G to be fulfilled and App fulfils G.

R3: $\neg cs(Ag, App1, App2)$
$\leftarrow h(Ag, App1), l(Ag, App1), \neg l(Ag, App2)$

Namely, Ag cannot swap $App1$ for $App2$ if, it is believed, Ag has and likes $App1$, and, it is assumed, Ag dislikes $App2$.

We could include additional inference rules, for example, for the case that an agent cannot swap an appointment it does not have or to allow agents to have multiple requirements, but the above three inference rules will be sufficient to demonstrate the AABA framework as is the focus of the paper.

The personal facts of agents x and y are as follows:

$$\mathcal{R}^p_x = \{h(x, app1),\ h(y, app2),\ r(x, seeDrAli),\ fr(app1, fridayApp), fr(app2, seeDrAli)\}$$
$$\mathcal{R}^p_y = \{h(y, app2),\ r(y, fridayApp),\ fr(app2, fridayApp), fr(app2, seeDrAli)\}$$

We assume here, for the sake of simplicity of demonstration, that agents have correct beliefs about the appointments allocated to themselves and other agents. In general, agents may have incorrect, out of date, or inconsistent beliefs. This will be the subject of future work. Our focus in this example are agents that have incomplete information. Indeed, e.g. y has no knowledge of x's appointments and goals.

5. Fact-supported Arguments

We generalise the notion of an argument that has a support consisting of assumptions only as in conventional ABA for our AABA framework. We define arguments as a deduction from a sentence p with a support consisting of assumptions *as well as* personal facts. Following (Dung, Kowalski, and Toni 2009), we define *fact-supported arguments* (abbreviated '*f-arguments*') for a claim p with support S as finite trees with p at the root and S at the leaves. Nodes in this tree are connected by the inference rules, with sentences matching the conclusion of an inference rule connected as parent nodes to sentences matching the premises of the inference rule as children nodes. The leaves in this tree are assumptions, personal facts or the special extra-logical symbol τ. We define first finite (deduction) trees and then f-arguments as a specific kind of such trees.

Definition 2 *Given an AABA framework $\mathcal{AF}_{Agent}$, a* finite (deduction) tree *$\mathcal{T}$ for $p \in \mathcal{L}$ (the* conclusion *or* claim*) is as follows:*

- *the root of $\mathcal{T}$ is labelled by p and denoted $root(\mathcal{T})$*
- *for every node N of $\mathcal{T}$*
  - *if N is a leaf then N is labelled by a sentence in $\mathcal{L}$ or by τ;*
  - *if N is not a leaf and l_N is the label of N, then there is an inference rule $l_N \leftarrow b_1, \ldots, b_m$ ($m \geq 0$) in $\mathcal{R}_c$ and either $m = 0$ and the child of N is labelled by τ*

    or $m > 0$ and N has m children, labelled by $b_1, \ldots, b_m$ (respectively)
- *the set of sentences, not including τ, labelling the leaves of $\mathcal{T}$ is denoted $leaves(\mathcal{T})$.*

Definition 3 *Given an AABA framework $\mathcal{AF}_{Agent}$, a* f-argument *for $p \in \mathcal{L}$ (the* conclusion *or* claim*) with* support *$S \subseteq \mathcal{A} \cup \mathcal{R}^p$ is a finite (deduction) tree $\mathcal{T}$ where $root(\mathcal{T}) = p$ and $leaves(\mathcal{T}) = S$.*

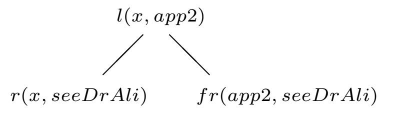

Figure 1: A f-argument for $l(x, app2)$ with support $\{r(x, seeDrAli),\ fr(app2, seeDrAli)\}$ wrt $\mathcal{AF}_x$ as defined in Section 4.

We illustrate in Figure 1 a finite tree that is a f-argument. F-arguments for a claim can be captured by means of backward deductions defined as follows.

Definition 4 *Given an AABA framework $\mathcal{AF}_{Agent}$ and a selection function* [1] *$f : 2^{\mathcal{L}} \rightarrow \mathcal{L}$, a* backward deduction *of $S \subseteq \mathcal{A} \cup \mathcal{R}^p$ for a claim $p \in \mathcal{L}$ is a finite sequence of tuples*

$$\langle C_0, S_0\rangle, \ldots, \langle C_i, S_i\rangle, \ldots, \langle C_n, S_n\rangle$$

where $n \geq 1$, $C_0 = \{p\}$, $S_0 = C_n = \emptyset$, $S_n = S$, and, for every $0 \leq i < n$, if $f(C_i) = \sigma$, then

1. *If σ is such that $\sigma \in \mathcal{R}^p$ or $\sigma \in \mathcal{A}$, then*
   - $C_{i+1} = C_i - \{\sigma\}$
   - $S_{i+1} = S_i \cup \{\sigma\}$
2. *Otherwise, choose $\sigma \leftarrow b_1, \ldots, b_m \in \mathcal{R}_c$ and*
   - $C_{i+1} = (C_i - \{\sigma\}) \cup \{b_1, \ldots, b_m\}$
   - $S_{i+1} = S_i$

We write $S \vdash^f_{\mathcal{AF}_{Agent}} p$ if there exists a backward deduction of S for p. [2]

Backward deductions are a generalisation of SLD resolution, which is the basis of proof procedures for logic programming. As in SLD, if there is a backward deduction using one selection function for picking sentences in C_i, then there is a backward deduction using any other selection function. Thus, different selection functions are simply different but equivalent ways of generating the same argument for a claim.

The following example illustrates the notion of backward deduction.

Example 1 *Given the AABA framework $\mathcal{AF}_x$ from Section 4, a backward deduction*

$$\{r(x, seeDrAli),\ fr(app2, seeDrAli)\} \vdash_{\mathcal{AF}_x} l(x, app2)$$

built by x affirming that it likes $app2$ is obtained as shown in Table 1. At step $i = 0$, $R2 \in \mathcal{R}_c$ is chosen in applying Case 2 of Definition 4. At step $i = 1$, Case 1 is applied since $r(x, seeDrAli) \in \mathcal{R}^p_x$. At step $i = 2$, Case 1 is applied since $fr(app2, seeDrAli) \in \mathcal{R}^p_x$.

In Definition 4, we use the word "choose" to identify backtrackable points in the search for a backward deduction. The need for this is illustrated by the following example:

Example 2 *Consider an AABA framework $\mathcal{AF}_{Agent}$ with*

$$\mathcal{L} = \{p, q, r\}, \mathcal{A} = \emptyset, \mathcal{R}_c = \{p \leftarrow q\} \cup \{p \leftarrow r\}, \mathcal{R}^p = \{r\}$$

[1] A selection function f takes as input a set of sentences C_i and returns a single element σ from C_i.

[2] We write $\vdash_{\mathcal{AF}_{Agent}}$ instead of $\vdash^f_{\mathcal{AF}_{Agent}}$ where the particular selection function is unimportant in the given context.

i	C_i	S_i	Apply…
0	$\{\underline{l(x, app2)}\}$	$\{\}$	Case 2
1	$\{\underline{r(x, seeDrAli)}, fr(app2, seeDrAli)\}$	$\{\}$	Case 1
2	$\{\underline{fr(app2, seeDrAli)}\}$	$\{r(x, seeDrAli)\}$	Case 1
3	$\{\}$	$\{r(x, seeDrAli), fr(app2, seeDrAli)\}$	

Table 1: Backward deduction for $l(x, app2)$ in $\mathcal{AF}_x$. The underlined atoms are picked by the selection function.

If in the search for a backward deduction for p, $p \leftarrow q \in \mathcal{R}_c$ is chosen when applying case 2 of Definition 4, then, since $q \notin \mathcal{R}^p$, $q \notin \mathcal{A}$ and there is no inference rule $q \leftarrow b_1, \ldots, b_m \in \mathcal{R}_c$, the search for a backward deduction needs to backtrack and choose $p \leftarrow r \in \mathcal{R}_c$, leading eventually to $\{r\} \vdash_{\mathcal{AF}_{Agent}} p$.

We end this section by proving the relationship between f-arguments and backward deductions.

Lemma 1 *If, for some selection function f, there is a backward deduction $S \vdash^f_{\mathcal{AF}_{Agent}} p$, then there is a f-argument for p with support S wrt $\mathcal{AF}_{Agent}$.*

Lemma 2 *If there is a f-argument for p with support S wrt $\mathcal{AF}_{Agent}$, then there is a backward deduction $S \vdash^f_{\mathcal{AF}_{Agent}} p$ for any selection function f.*

The proofs of Lemmas 1 and 2 can be found in an accompanying technical report (see Section 9).

6. Computation of Admissible Arguments

The AB-dispute derivations (which we abbreviate '*dd*s') of (Dung, Mancarella, and Toni 2007) can be used to construct an admissible set of assumptions supporting a claim. We generalise this for AABA frameworks and present a new computational model, which we call *fact-inclusive AB-dispute derivations* (abbreviated '*fdd*s'), that constructs an admissible defence for a claim as well as personal facts used during the construction. These derivations, similar to *dd*s, can be seen as a game between two fictional players - a proponent and an opponent. The rules of the game are roughly as follows: the proponent puts forward a f-argument for a claim; the opponent disputes the proponent's argument by presenting f-arguments attacking assumptions of the proponent's argument; the proponent in turn defends its argument by counter-attacking the opponent's attacks with other f-arguments, which the opponent in turn attacks, and so on, until the proponent has defended itself against all attacks.

As for *dd*s, in conducting this game, the proponent cannot attack any of its own assumptions, nor does it need to counter-attack any assumption it has attacked previously, nor does it need to defend any assumption it has already defended. Differently from *dd*s, in *fdd*s, f-arguments are fully constructed before being attacked. Thus only actual f-arguments are manipulated (rather than "potential", see (Gaertner and Toni 2008)). Also, in *fdd*s unlike *dd*s, the proponent keeps track of personal facts used in its arguments. Together, the assumptions and personal facts used by the proponent make up its defence for a claim.

We define a *fdd* as a sequence of tuples of the form $\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle$. Such a tuple represents the state of the derivation at the ith step and is to be read as follows: $\mathcal{P}_i$ is a set of sentences, the claims of the proponent to be expanded and defended; $\mathcal{O}_i$ are the sets consisting of personal facts and assumptions that support attacks of the opponent on the assumptions of the proponent, and each such set is to be attacked by the proponent; $\mathcal{D}_i$ (*Defences*) is the set of personal facts and assumptions encountered so far in the derivation sequence that support the claims of the proponent; $\mathcal{C}_i$ (*Culprits*) is the set of assumptions of the opponent that have already been attacked by the proponent.

Definition 5 *Given an AABA framework $\mathcal{AF}_{Agent}$, and selection functions $f : 2^{\mathcal{L}} \rightarrow \mathcal{L}$ and $g : 2^{2^{\mathcal{L}}} \rightarrow 2^{\mathcal{L}}$,* [3] *a* fdd *of a defence set $\mathcal{D} \subseteq \mathcal{A} \cup \mathcal{R}^p$ for a set of sentences $\mathcal{P} \subseteq \mathcal{L}$ is a finite sequence of tuples*

$$\langle \mathcal{P}_0, \mathcal{O}_0, \mathcal{D}_0, \mathcal{C}_0 \rangle, \ldots, \langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle, \ldots, \langle \mathcal{P}_n, \mathcal{O}_n, \mathcal{D}_n, \mathcal{C}_n \rangle$$

where $\mathcal{P}_0 = \mathcal{P}$, $\mathcal{D}_0 = \mathcal{P} \cap (\mathcal{A} \cup \mathcal{R}^p)$, $\mathcal{O}_0 = \mathcal{C}_0 = \emptyset$, $\mathcal{P}_n = \mathcal{O}_n = \emptyset$, $\mathcal{D}_n = \mathcal{D}$, and, for every $0 \leq i < n$, only one $f(\mathcal{P}_i) = \sigma$ or one $g(\mathcal{O}_i) = \Sigma$ is picked [4] *and*

1. *If $f(\mathcal{P}_i) = \sigma$ is picked, then*
   - *(a) if $\sigma \in \mathcal{R}^p$, then*
     - $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$
     - $\mathcal{O}_{i+1} = \mathcal{O}_i$
     - $\mathcal{D}_{i+1} = \mathcal{D}_i$
     - $\mathcal{C}_{i+1} = \mathcal{C}_i$
   - *(b) if $\sigma \in \mathcal{A}$, then*
     - $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$
     - $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{S \mid \sigma' \in c(\sigma), S \vdash^f_{\mathcal{AF}_{Agent}} \sigma', S \cap \mathcal{C}_i = \emptyset\}$ *(filtering of culprits by culprits)*
     - $\mathcal{D}_{i+1} = \mathcal{D}_i$
     - $\mathcal{C}_{i+1} = \mathcal{C}_i$
   - *(c) if $\sigma \notin \mathcal{A} \cup \mathcal{R}^p$, choose a backward deduction $S \vdash^f_{\mathcal{AF}_{Agent}} \sigma$ such that $S \cap \mathcal{C}_i = \emptyset$ (filtering of defences by culprits), then*
     - $\mathcal{P}_{i+1} = (\mathcal{P}_i - \{\sigma\}) \cup ((S \cap \mathcal{A}) - \mathcal{D}_i)$ *(filtering of defences by defences)*
     - $\mathcal{O}_{i+1} = \mathcal{O}_i$
     - $\mathcal{D}_{i+1} = \mathcal{D}_i \cup S$
     - $\mathcal{C}_{i+1} = \mathcal{C}_i$
2. *If $g(\mathcal{O}_i) = \Sigma$ is picked, then*
   - *(a) if $\Sigma \cap \mathcal{C}_i \neq \emptyset$ (filtering of culprits by culprits), then*
     - $\mathcal{P}_{i+1} = \mathcal{P}_i$
     - $\mathcal{O}_{i+1} = \mathcal{O}_i - \{\Sigma\}$
     - $\mathcal{D}_{i+1} = \mathcal{D}_i$
     - $\mathcal{C}_{i+1} = \mathcal{C}_i$

[3] f takes as input a set of sentences $\mathcal{P}_i$ and returns a single element $\sigma \in \mathcal{P}_i$, while g takes as input a set of sets of sentences $\mathcal{O}_i$ and returns a single element $\Sigma \in \mathcal{O}_i$.

[4] Depending on the player choice, as discussed later.

(b) if $\Sigma \cap \mathcal{C}_i = \emptyset$, choose $\sigma \in \Sigma$ such that $\sigma \in \mathcal{A}$ and $\sigma \notin \mathcal{D}_i$ (filtering of culprits by defences), and choose $\sigma' \in c(\sigma)$, then
- $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\sigma'\}$
- $\mathcal{O}_{i+1} = \mathcal{O}_i - \{\Sigma\}$
- $\mathcal{D}_{i+1} = \mathcal{D}_i \cup (\{\sigma'\} \cap (\mathcal{A} \cup \mathcal{R}^p))$
- $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{\sigma\}$

We write $\mathcal{D} \models^{f,g}_{\mathcal{AF}_{Agent}} \mathcal{P}$ if there exists a fdd *of $\mathcal{D}$ for $\mathcal{P}$. We write $\mathcal{D} \models^{f,g}_{\mathcal{AF}_{Agent}} p$ if $\mathcal{P} = \{p\}$.* [5]

Similarly to Definition 4, we use the word "choose" in Definition 5 above to identify backtrackable points in the search for a *fdd*.

Intuitively, two choices (at least) 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 ($\mathcal{O}_i$). If the proponent is chosen, then a sentence σ in $\mathcal{P}_i$ is picked and one of three cases (1a, 1b or 1c) apply. If the opponent is chosen, then a set Σ in $\mathcal{O}_i$ is picked and one of two cases (2a or 2b) apply.

In case 1a, the proponent plays and σ is a personal fact: this is simply removed from the proponent's set. Indeed, σ will have been added to the defence set previously.

In case 1b, the proponent plays and σ is an assumption: this is removed from the proponent's set and all (suppports of) f-arguments that attack σ and contain no culprit assumptions are added to the opponent's set.

In case 1c, the proponent plays and σ is neither a personal fact nor an assumption: σ is removed from the proponent's set and, choosing a support S for σ that contains no culprit assumptions, replaced by the assumptions of S after the defence assumptions have been filtered.

In case 2a, the opponent plays and Σ contains a culprit assumption: Σ is simply removed from the opponent's set, as it has already been counter-attacked.

In case 2b, the opponent plays and Σ contains no culprit assumptions: Σ is removed from the opponent's set. A non-defence (trivially non-culprit) assumption σ in Σ is chosen and added to the set of culprits. Also, a contrary σ' of σ is chosen, added to the proponent's set and, if σ' is a personal fact or assumption, also added to the defence set.

We demonstrate in Examples 3 and 4 below a successful and a failed (respectively) *fdd*. Here and in the remainder of the paper, "*fdd* of $\mathcal{D}$ for p" will stand for "*fdd* of $\mathcal{D}$ for $\{p\}$".

Example 3 *Given the AABA framework $\mathcal{AF}_x$ of Section 4, a* fdd *$S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ is obtained as shown in Table 2. The* fdd *here coincides with the f-argument. At step $i = 0$, $sa(x, y, app1, app2) \in \mathcal{P}_0$ is picked and $S \vdash_{\mathcal{AF}_x} sa(x, y, app1, app2)$ chosen in applying Case 1c. At step $i = 1$, $\neg l(x, app1) \in \mathcal{P}_1$ is picked and Case 1b applied. Note that $c(\neg l(x, app1)) = \{l(x, app1)\}$ and $l(x, app1)$ has no support in $\mathcal{AF}_x$, hence $\mathcal{O}_2 = \mathcal{O}_1$. At step $i = 2$, $cs(y, app2, app1) \in \mathcal{P}_2$ is picked and Case 1b applied. Note that $c(cs(y, app2, app1)) = \{\neg cs(y, app2, app1)\}$ and $\neg cs(y, app2, app1)$ has no support in $\mathcal{AF}_x$, hence $\mathcal{O}_3 = \mathcal{O}_2$. Note that S includes facts to be communicated, which would not be accumulated by the* dd *procedure.*

[5] We write $\models_{\mathcal{AF}_{Agent}}$ instead of $\models^{f,g}_{\mathcal{AF}_{Agent}}$ where the particular selection functions are unimportant in the given context.

i	$\mathcal{P}_i$	$\mathcal{O}_i$	$\mathcal{D}_i$	$\mathcal{C}_i$	Case…
0	$\{\underline{sa(x, y, app1, app2)}\}$	$\{\}$	$\{\}$	$\{\}$	1c
1	$\{\underline{\neg l(x, app1)},$	$\{\}$	S	$\{\}$	1b
	$cs(y, app2, app1)\}$				
2	$\{\underline{cs(y, app2, app1)}\}$	$\{\}$	S	$\{\}$	1b
3	$\{\}$	$\{\}$	S	$\{\}$	

Table 2: *fdd* for $sa(x, y, app1, app2)$ in $\mathcal{AF}_x$. $S = \{h(x, app1),\ r(x, seeDrAli),\ fr(app2, seeDrAli),\ \neg l(x, app1),\ cs(y, app2, app1)\}$. The underlined atoms are picked by f.

i	$\mathcal{P}_i$	$\mathcal{O}_i$	$\mathcal{D}_i$	$\mathcal{C}_i$	Case
0	$\{\underline{sa(x, y, app1, app2)}\}$	$\{\}$	$\{\}$	$\{\}$	1c
1	$\{\neg l(x, app1),$	$\{\}$	S	$\{\}$	1b
	$\underline{cs(y, app2, app1)}\}$				
2	$\{\neg l(x, app1)\}$	$\{\underline{S'}\}$	S	$\{\}$	2b
3	$\{\neg l(x, app1),$	$\{\}$	S	$\{\neg l(y, app1)\}$	
	$\underline{l(y, app1)}\}$				

Table 3: Attempt to build a *fdd* for $sa(x, y, app1, app2)$ in $\mathcal{AF}_y$. S is as defined in Table 2. $S' = \{has(y, app2), r(y, fridayApp), fr(app2, fridayApp), \neg l(y, app1)\}$. The underlined elements are picked by f and g.

Example 4 *Given the AABA framework $\mathcal{AF}_y$ of Section 4 but such that additionally $h(x, app1), r(x, seeDrAli) \in \mathcal{R}^p_y$, a* fdd *for $sa(x, y, app1, app2)$ cannot be found as shown in Table 3. At step $i = 0$, $sa(x, y, app1, app2) \in \mathcal{P}_0$ is picked and $S \vdash_{\mathcal{AF}_y} sa(x, y, app1, app2)$ chosen in applying Case 1c of Definition 5. At step $i = 1$, $cs(y, app2, app1) \in \mathcal{P}_1$ is picked and Case 1b applied. Note that $c(cs(y, app2, app1)) = \{\neg cs(y, app2, app1)\}$ and S' is the only support for $\neg cs(y, app2, app1)$ in $\mathcal{AF}_y$, hence $\mathcal{O}_2 = \mathcal{O}_1 \cup \{S'\}$. At step $i = 2$, $S' \in \mathcal{O}_2$ is picked and in applying Case 2b, (the only assumption) $\neg l(y, app1) \in S'$ and (the only contrary) $l(y, app1) \in c(\neg l(y, app1))$ chosen. At step $i = 3$, picking $l(y, app1) \in \mathcal{P}_3$, the* fdd *fails since none of the cases 1a to 1c are applicable for $l(y, app1)$ according to $\mathcal{AF}_y$. In particular, $l(y, app1) \notin \mathcal{R}^p_y$ nor is there a support for $l(y, app1)$ in $\mathcal{AF}_y$. Further, there are no points in the derivation sequence where backtracking can be successfully applied.*

7. An Application: Arguments in Dialogue

*Fdd*s, giving $\mathcal{D} \models_{\mathcal{AF}_{Agent}} p$, can be seen as a means of generating "communicable arguments" for claims p to be exchanged between communicating agents. Note that typically these are not f-arguments as they could include in $\mathcal{D}$ additional facts and assumptions needed to defend the claim against attacks. We refer to these "arguments" as *fdd arguments*.

We demonstrate in this section how exchanging fdd arguments between agents can be useful in dialogue. Note that this section is solely for illustration and not intended to define a dialogue game for argumentation. Whilst frameworks have been presented for argumentation-based dialogue games, e.g. (Prakken 2005), our work is orthogonal and presents instead a means of representing and generating arguments that could be fed into such frameworks.

Given the AABA frameworks $\mathcal{AF}_x$ and $\mathcal{AF}_y$ of Section 4 and ignoring details of the communication protocol and policy, we demonstrate how, by exchanging arguments, agents x and y could come to an agreement that $sa(x, y, app1, app2)$ is an acceptable action. Informally, the dialogue between the two agents proceeds as follows:

x: We should swap $app1$ and $app2$ because …
 … and I assume you can swap $app2$ for $app1$.
y: I cannot swap $app2$ for $app1$ because …
 … and I assume $app1$ is no good for me.
x: Ah, but $app1$ is good for you because …,
 hence we should swap $app1$ and $app2$.
y: Ok. I agree to the swap.

We show in Table 4 and explain here the fdd arguments uttered by the agents and the beliefs each agent adds to its personal facts as a result of the dialogue. The dialogue takes place over time points 1–4. Time 0 is prior to the dialogue.

t=0: There is a *fdd* $S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ (as derived in Table 2). However, there is no *fdd* for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$.

t=1: x communicates its fdd argument $S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ to y and y revises $\mathcal{R}^p_y$ to include $has(x, app1)$ and $r(x, seeDrAli)$, given in S. Then, y knows what x has and requires.

t=2: y can now build $S \vdash_{\mathcal{AF}_y} sa(x, y, app1, app2)$. However, a *fdd* for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$ still does not exist, as shown in Table 3. The successfully attacking argument in the failed *fdd* sequence is $S' \models_{\mathcal{AF}_y} \neg cs(y, app2, app1)$. Then, y communicates this fdd argument to x and x revises $\mathcal{R}^p_x$ to include $r(y, fridayApp)$ and $fr(app2, fridayApp)$, given in S'.

t=3: x can now build a f-argument $S' \vdash_{\mathcal{AF}_x} \neg cs(y, app2, app1)$. However, this is not an acceptable argument in $\mathcal{AF}_x$ since the (only) assumption $\neg l(y, app1)$ in S' has a contrary $l(y, app1)$ for which there is an acceptable *fdd* $S'' \models_{\mathcal{AF}_x} l(y, app1)$. By means of this defending argument, a *fdd* $S \cup S'' \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ exists. Then x communicates this fdd argument to y and y revises $\mathcal{R}^p_y$ to include $fr(app1, fridayApp)$, given in S''.

t=4: y is now able to build a f-argument $S'' \vdash_{\mathcal{AF}_y} l(y, app1)$ as well as an acceptable *fdd* for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$. Hence the agents end in agreement.

t	x utters…	y utters…
1	$S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$	
2		$S' \models_{\mathcal{AF}_y} \neg cs(y, app2, app1)$
3	$S''' \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$	
4		$S''' \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$

t	$\mathcal{R}^p_x$ plus…	$\mathcal{R}^p_y$ plus…
1		$has(x, app1), r(x, seeDrAli)$
2	$r(y, friApp)$,$fr(app2, friApp)$	
3		$fr(app1, fridayApp)$
4		

Table 4: Arguments exchanged between x and y regarding the acceptability of $sa(x, y, app1, app2)$, and respective changes to each agent's personal facts. S and S' are as defined in Tables 2 and 3 (resp). S'' = $\{r(y, fridayApp), fr(app1, fridayApp)\}$. S''' = $S \cup S''$.

8. Theoretical Results

In this section, we state the following main results for any ABA framework $\mathcal{AF}$ and corresponding AABA framework $\mathcal{AF}_{Agent}$:

- *fdd*s correspond to (a variant of) the *dd*s of (Dung, Mancarella, and Toni 2007) (see Theorems 1 and 2, and Corollary 1);
- *fdd*s are correct in that the assumptions in the support of the input claims they compute is indeed admissible (see Corollary 2);
- *fdd*s are complete for p-acyclic ABA frameworks (see Corollary 3).

The variant of *dd*s we use is as defined in (Gaertner 2009). The cases 2ic.1 and 2ic.2 of the *dd*s of (Dung, Mancarella, and Toni 2007) are combined into one single case 2ic which states:

$\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{c(\sigma)\}$; $\mathcal{O}_{i+1} = \mathcal{O}_i - S$;
$\mathcal{D}_{i+1} = \mathcal{D}_i \cup (\mathcal{A} \cap \{c(\sigma)\})$; $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{\sigma\}$.

The variant is proven in (Gaertner 2009) to be correct, and to be 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 (Dung, Mancarella, and Toni 2007). We prove our results for ABA frameworks where the contrary of every assumption is a singleton set. This can be done without loss of generality since there is a one-to-one correspondence between such ABA frameworks and ABA frameworks with multiple contraries (Gaertner and Toni 2008). When $c(\sigma)$ is a singleton set $\{\sigma'\}$, we denote $c(\sigma)$ simply as σ', i.e. $c(\sigma) = \sigma'$.

Theorem 1 *If, using some selection functions f and g, $\mathcal{D} \models^{f,g}_{\mathcal{AF}_{Agent}} p$, then, using some selection function f', there is a* dd *of $\mathcal{D} \cap \mathcal{A}$ for p.*

Theorem 2 *If, using some selection function f, there is a* dd *of a defence set $A \subseteq \mathcal{A}$ for a claim p, then, for some selection function g, $A \cup F \models^{f,g}_{\mathcal{AF}_{Agent}} p$ for some $F \subseteq \mathcal{R}^p$.*

The proofs of Theorems 1 and 2 can be found in an accompanying technical report (see Section 9).

Corollary 1 *If $\mathcal{R}^p = \emptyset$, then there is a* fdd *of a defence set $\mathcal{D}$ for p iff there is a* dd *of $\mathcal{D}$ for p.*

The proof of Corollary 1 follows straightforwardly from Theorems 1 and 2 above.

Corollary 2 *For every* fdd *of a defence set $\mathcal{D} = A \cup F$ ($A \subseteq \mathcal{A}$, $F \subseteq \mathcal{R}^p$) for $p \in \mathcal{L}$, the set of assumptions A is admissible and there exist some $A' \subseteq A$ and $F' \subseteq F$ such that $A' \cup F' \vdash_{\mathcal{AF}_{Agent}} p$.*

The proof of Corollary 2 follows straightforwardly from Theorem 1 above and the soundness theorem for *dd*s (i.e.

Theorem 4.3 of (Dung, Mancarella, and Toni 2007), as formulated in (Gaertner 2009)).

We define a *positively acyclic* (or *p-acyclic* for short) ABA framework as in (Dung, Mancarella, and Toni 2007). By $\mathcal{AF}^+$ we denote the ABA framework obtained by deleting all assumptions appearing in the premises of the inference rules of $\mathcal{AF}$. The *dependency graph* of $\mathcal{AF}^+$ is a directed graph where:

- the nodes are the atoms occurring in $\mathcal{AF}^+$;
- a (directed) arc from a node p to a node q is in the graph if and only if there exists an inference rule $p \leftarrow B$ in $\mathcal{AF}^+$ such that q occurs in B.

Definition 6 *An ABA framework $\mathcal{AF}$ is* p-acyclic *if the dependency graph of $\mathcal{AF}^+$ is acyclic. An AABA framework $\langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}^p, \mathcal{A}, c\rangle$ is* p-acyclic *iff $\langle \mathcal{L}, \mathcal{R}_c \cup \mathcal{R}^p, \mathcal{A}, c\rangle$ is* p-*acyclic.*

Definition 7 *A* partial deduction *for a claim p, given a selection function f, is a sequence $\langle C_0, S_0\rangle, \ldots, \langle C_j, S_j\rangle$ where $j \geq 0$, $C_0 = \{p\}$, $S_0 = \emptyset$ and each $\langle C_{i+1}, S_{i+1}\rangle$ ($0 \leq i < j$) is constructed from $\langle C_i, S_i\rangle$ by picking $f(C_i) = \sigma$ and applying case 1 or 2 of Definition 4.*

Theorem 3 *Given a p-acyclic framework, there exists no infinite partial deduction.*

The proof of Theorem 3 is trivial since there is a well-ordering of all sentences in the language of the p-acyclic framework, i.e. whenever a sentence belongs to the premise of an inference rule, then the sentence is lower in ordering than the conclusion of the inference rule.

Theorem 3 guarantees that the search for a f-argument always terminates (either successfully, finding such an argument, or not). In the case of p-acyclic frameworks with a finite underlying language $\mathcal{L}$ the *fdd*s are complete, in the following sense:

Corollary 3 *Let $\mathcal{AF}_{Agent}$ be a p-acyclic AABA framework such that $\mathcal{L}$ is finite. Then, for each sentence $l \in \mathcal{L}$, if l is an admissible claim then*

- *there exists a* fdd *for l;*
- *for each admissible set $A \subseteq \mathcal{A}$, if, for some $A' \subseteq A$ and $F \subseteq \mathcal{R}^p$, $A' \cup F \vdash_{\mathcal{AF}_{Agent}} l$, then there is a* fdd *$\mathcal{D} \models_{\mathcal{AF}_{Agent}} l$ such that $A' \cup F \subseteq \mathcal{D}$.*

The proof of Corollary 3 follows straightforwardly from Theorem 2 above and the completeness theorem for *dd*s for p-acyclic ABA frameworks (i.e. Theorem 4.4 of (Dung, Mancarella, and Toni 2007)).

9. Conclusion

We have proposed an extension to the ABA framework (Dung, Kowalski, and Toni 2009) for multi-agent system settings where agents have some common knowledge but may also have some personal knowledge not necessarily shared by all agents. This is complementary of work in (Hunter 2007). In exchanging arguments between one another, agents in such an agent system would need to include personal beliefs that the recipient agent may be unaware of. Such arguments can be seen as enthymemes (Hunter 2007).

We have proposed a revised notion of ABA argument that accumulates an agent's personal beliefs as well as its assumptions. Accordingly, we have proposed a computational model for generating admissible arguments, called fact-inclusive AB-dispute derivations (*fdd*s), which accumulates an agent's personal facts as well as its assumptions. We have demonstrated how such outcomes of *fdd*s could be usefully exchanged between agents in dialogue. We plan to generalise our AABA framework and the associated computation models to allow in $\mathcal{R}^p$ general inference rules rather than just facts. Also, we plan to apply our framework and the exchange of arguments in the context of concrete inter-agent communication. We also aim to address situations where agents have inconsistent (as well as incomplete) beliefs.

We have implemented the backward deduction and *fdd* procedures in Prolog, and have used these in the implementation of the negotiation policies of (Hussain and Toni 2008). The implementations of the procedures and negotiation policies, as well as the accompanying technical report, can be downloaded from www.doc.ic.ac.uk/∼ah02.

References

Amgoud, L.; Maudet, N.; and Parsons, S. 2000. Modelling dialogues using argumentation. In *ICMAS'00*. IEEE Press.

Amgoud, L.; Parsons, S.; and Maudet, N. 2000. Arguments, dialogue, and negotiation. In *ECAI'00*, 338–342. IOS Press.

Black, E., and Hunter, A. 2008. Using enthymemes in an inquiry dialogue system. In *AAMAS'08*, 437–444. ACM Press.

Dung, P. M.; Kowalski, R. A.; and Toni, F. 2009. Assumption-based argumentation. In *Argumentation in Artificial Intelligence*. Springer. 199–218.

Dung, P. M.; Mancarella, P.; and Toni, F. 2007. Computing ideal sceptical argumentation. *Artificial Intelligence* 171(10–15):642–674.

Gaertner, D., and Toni, F. 2008. Hybrid argumentation and its properties. In *COMMA'08*, 183–195. IOS Press.

Gaertner, D. 2009. *Argumentation and Normative Reasoning*. Ph.D. Dissertation, Imperial College.

Hunter, A. 2007. Real arguments are approximate arguments. In *AAAI'07*, 66–71. MIT Press.

Hussain, A., and Toni, F. 2008. On the benefits of argumentation for negotiation (preliminary version). In *EUMAS'08*.

Mcburney, P.; Hitchcock, D.; and Parsons, S. 2002. The eightfold way of deliberation dialogue. *International Journal of Intelligent Systems* 22(1):95–132.

Prakken, H. 2005. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation* 15(6):1009–1040.

Rahwan, I.; Ramchurn, S. D.; Jennings, N. R.; McBurney, P.; Parsons, S.; and Sonenberg, L. 2003. Argumentation-based negotiation. *The Knowledge Engineering Review* 18(4):343–375.