On the benefits of argumentation for negotiation - preliminary version

Adil Hussain and Francesca Toni
Department of Computing, Imperial College London,
180 Queen's Gate, London SW7 2AZ, UK
{ah02, ft}@doc.ic.ac.uk

December 4, 2008

Abstract

We present preliminary work on the benefits of argumentation-based negotiation, for a simple framework for one-to-one negotiation between agents in a resource reallocation setting. Agents engage in dialogues with other agents in order to obtain resources they need but do not have. Dialogues are regulated by simple communication policies that allow agents to provide reasons (arguments) for their refusals to give away resources; agents use assumption-based argumentation in order to deploy these policies. We assess the benefits of providing these reasons both informally and experimentally: by providing reasons, agents are "more effective" in identifying a reallocation of resources if one exists and failing if none exists.

1 Introduction

The need for agents to interact arises in many settings. One such setting is the resource reallocation problem, where resources are limited, controlled and required by agents interacting with one another. This setting arises naturally, for example, in electronic commerce (where resources are commodities equipped with prices) and the grid (where resources are computational entities equipped with computational power).

The inter-agent interaction method of choice in this report is that of dialogues, namely orderly sequences of message exchanges between two agents, where each agent has a goal and the dialogues, as a whole, also have a goal, notably that of solving a resource reallocation problem for all dialogue participants. Concretely, we focus on *negotiation* dialogues, whereby agents try to arrive at a decision concerning possession of resources. We adopt a "generative" approach to negotiation, by providing constructive *policies* for the automatic generation of dialogues by autonomously reasoning agents. We consider two policies and, as a result, two forms of negotiation: the first is a trivial adaptation of the policy of

[9], whereby agents simply agree to release resources they have but do not need, and refuse otherwise; the second is a slightly improved version, whereby agents provide reasons (arguments) for their refusal. Argumentation-based negotiation is a particular class of negotiation, the claim for which is that allowing agents to provide arguments and justifications as part of the negotiation process, increases the likelihood and/or speed of agreements being reached [7]. We focus in this report on substantiating this claim, by studying the use of argumentation for improved effectiveness of the negotiation process, in particular concerning the number of dialogues and dialogue moves that need to be performed without affecting the quality of solutions reached.

Whilst there has been discussion on protocols for argumentation-based negotiation (e.g. [10]), little work exists on the agents' decision making models to support argumentative negotiation, i.e. how the agents would be built and the strategic reasoning that is to go on in the agents' minds to determine the best course of action at any given time. To the best of our knowledge, the only work aiming at a formal implementable model of argumentation-based negotiation is that of [3], but even there the embedded persuasion dialogues are left unspecified for future work. We define formally an argumentation framework to support the agent decision-making, underlying the use of both policies. This framework is a concrete instance of an existing general-purpose argumentation framework, known as Assumption-Based Argumentation (ABA) [1, 4].

The report is structured as follows: In Section 2 we define the key components of our framework, amounting to the notion of agent system, agent's mind, resource reallocation problem and dialogues. In Section 3 we provide the core representation of the agents' mind in ABA, for decision-making prior and during dialogues. In Section 4 we present the two agent policies, the very simple one and the reason-based one. In Section 5 we analyse, informally, properties of both policies. Lastly, in Section 6, we conclude, discuss the implementation of the policies and present some possible directions for future work. This report is an expansion of [5]. In particular, this report includes details of proofs of results and experiments, as detailed in appendices A, B and C.

2 Preliminaries

Below we will adopt the following notation: $\neg$ stands for (classical) negation; terms beginning with a capital letter are variables; terms beginning in small-case are constants; _ stands for an anonymous variable (as in Prolog).

We consider agent systems where (i) each agent may have in its possession some (or no) resources; and (ii) each agent may need some (or no) resources, some of which it may already have and some of which it may not.

Definition 1 (Agent System) *An agent system is a (finite) set A, where each $x \in A$ is a constant, representing the name of an agent, equipped with a (finite, possibly empty) set of* allocated resources *$Res(x)$, a (finite, possibly empty) set of* needed resources *$G(x)$ (the* goal *of agent x) and a* belief base *$B(x)$.*

We will assume that resources are **indivisible** (i.e. agents cannot receive fractions of resources), **non-sharable** (i.e. a resource cannot be allocated to two or more agents at the same time), and **single-unit** (there is only one copy of each resource). Formally, given an agent system A, for every $x, y \in A$, $x \neq y$:

- $Res(x) \subseteq Res$ for some finite set Res of constants, and
- $Res(x) \cap Res(y) = \{\}$.

Naturally, we require that $A \cap Res = \{\}$ (namely, the same constant cannot be used to refer to an agent and a resource). Moreover, we assume that, given an agent system A, for every $x \in A$ it holds that $G(x) \subseteq Res$. Thus, Res is the set of resources *available* and *needed* in the agent system.

Example 1 *An agent system consisting of two agents $ag1$ and $ag2$, such that $ag1$ has $r1$ and needs $r2$, and $ag2$ has $r2$ and needs $r3$, is represented as follows: $A = \{ag1, ag2\}$, $Res(ag1) = \{r1\}$, $G(ag1) = \{r2\}$, $Res(ag2) = \{r2\}$, $G(ag2) = \{r3\}$, and $Res = \{r1, r2, r3\}$.*

Informally, the belief base of an agent is composed of (i) the names of the agents in the system, including its own; (ii) (some of) the resources that (some of) the other agents are allocated; (iii) (some of) the resources that (some of) the other agents need; and (iv) possibly, a belief that the agent's goal (of obtaining all resources it needs) cannot be achieved. Thus, in general, agents have only partial knowledge of which resources the other agents in the system are allocated and need. However, agents are aware of the existence of all their fellow agents in the agent system. We give a formal representation of agents' beliefs in Section 3.

Definition 2 (Resource Reallocation Problem - r.r.p.) *Given an agent system A as in definition 1, with each agent $x \in A$ equipped with a set of allocated resources $Res(x)$, a set of needed resources $G(x)$, and a belief base $B(x)$,*

- *the r.r.p. for an agent $x \in A$ is the problem of finding an agent system A' with $x \in A'$ and x is equipped in A' with a set of allocated resources $Res'(x) \supseteq G(x)$; we say that this A' solves the r.r.p. for x;*
- *the r.r.p. for the overall agent system A is the problem of finding an agent system A' solving the r.r.p. for every agent in A; we say that this A' solves the r.r.p. for A; we also say that the r.r.p. for A is solved if such A' is found.*

The set of agents A' solving a r.r.p. for A may contain agents other than the ones originally in A: these would typically be agents bringing in new resources needed by some agents in A but not available there. In this report, however, we will always obtain A' with the same agents as in A (but these agents will typically have different beliefs and allocated resources). Thus, given an agent system A, the r.r.p. for A can only be solved if

- for all $x \in A$, $G(x) \subseteq \bigcup_{y \in A} Res(y)$, namely all needed resources are available in the system; and

- there are no agents $x, y \in A$, $x \neq y$, such that $G(x) \cap G(y) \neq \{\}$, namely no two agents need the same resource (remember that resources are single-unit).

Note that we implicity assume, in the earlier definition 2, that the agents' needs ($G(x)$) are fixed and cannot change during the reallocation.

Example 2 *Given the agent system in example 1, a solution to the r.r.p. for $ag1$ exists, since $ag2$ has the resource $r2$ needed by $ag1$ (i.e. $G(ag1) \subseteq Res(ag1) \cup Res(ag2)$). However, no solution to the r.r.p. for $ag2$ exists, since the resource $r3$ needed by $ag2$ is not held by any agent in the system (i.e. $G(ag2) \not\subseteq Res(ag1) \cup Res(ag2)$). Consider instead an agent system as in example 1 but with an additional agent $ag3$ as follows: $A = \{ag1, ag2, ag3\}$, $G(ag3) = \{\}$, $Res(ag3) = \{r3\}$ and $Res = \{r1, r2, r3\}$. Now, a solution to the r.r.p. for $ag1$ still exists, but in addition a solution to the r.r.p. for $ag2$ also exists, since $G(ag2) \subseteq Res(ag1) \cup Res(ag2) \cup Res(ag3)$. Likewise for $ag3$. Thus, in this case, a possible solution to the r.r.p. for the overall agent system A is $A' = \{ag1, ag2, ag3\}$ such that $Res'(ag1) = \{r1, r2\}$, $Res'(ag2) = \{r3\}$, $Res'(ag3) = \{\}$ and $G(ag1)$, $G(ag2)$, $G(ag3)$, Res are as before.*

We will describe in Section 4 two simple agent policies to generate dialogues between agents for the reallocation of resources solving the above-defined resource reallocation problem. We will see that, as a result of these dialogues, the agents' set of allocated resources and belief base undergo changes. In the remainder of the report we will assume as given a set of resources Res and an agent system A with at least two agents.

Definition 3 (Dialogue Move) *A* dialogue move *is an instance of the schema* `tell(X,Y,Subject)`*, where* `X`$\in A$ *is the utterer,* `Y`$\in A$ *is the receiver,* `X`$\neq$`Y`*, and* `Subject` *is the content of the* dialogue move*, expressed in some given* content language.

The content language is assumed to be shared by all agents in the given agent system A. In our setting, the content language allows to describe: an agent attempting to obtain a needed resource, for a *reason*, by requesting it from another agent and waiting for a response; this could be an acceptance or a refusal to give away the requested resource, accompanied by a *reason*.

Definition 4 (Content Language) *The* content language *we consider consists of all instances of the following schemata (for* `R` $\in$ *Res):* `request(give(R)) because Reasons`*;* `accept(give(R))`*;* `refuse(give(R)) because Reasons`*.*

In Section 4 we will see that `Reasons` is a (possibly empty) set including some of the utterer's beliefs as well as information about its and others' allocated and needed resources, which may be useful for the receiver to know.

The dialogue moves obtained with our chosen content language are split into sets, as follows:

Definition 5 (Dialogue Initial/Final Move) *Any instance of the schema* `tell(X,Y,(request(give(R)) because Reasons))` *is a* dialogue initial move (regarding `R`)*. Any instance of the schemata* `tell(X,Y,accept(give(R)))` *or* `tell(X,Y,(refuse(give(R)) because Reasons))` *is a* dialogue (successful or unsuccessful*, respectively)* final move (regarding `R`)*.*

Dialogues are composed of dialogue moves regarding the same resource, beginning with a dialogue initial move and ending with a dialogue final move:

Definition 6 ((Successful/Unsuccessful) Dialogue Instance) *A* dialogue instance *wrt an initiating agent* x*, a responding agent* y *and a resource* r *is a sequence of two dialogue moves such that*

1. x *utters to* y *a* dialogue initial move *regarding* r*, and subsequently*

2. y *utters to* x *a* dialogue final move *regarding* r*.*

A successful dialogue instance *is a dialogue instance that ends with a* successful final move*. An* unsuccessful dialogue instance *is a dialogue instance that ends with an* unsuccessful final move*.*

A successful dialogue instance results in a resource being given by one agent to another. In general, an agent may need to initiate a number of dialogue instances with a number of agents before a particular resource is obtained or before it gives up its intention of obtaining the resource.

Definition 7 (Dialogue Sequence) *A* dialogue sequence *wrt an agent* x *and a resource* r *is a sequence of* dialogue instances *all initiated by* x *requesting* r *and such that the only successful dialogue instance in the sequence, if any, is the last one in the sequence. A dialogue sequence* terminates successfully *if it is finite and the last dialogue instance is successful. A dialogue sequence* terminates unsuccessfully *if it is finite and the last dialogue instance is unsuccessful.*

Note that we implicitly allow an agent to keep on asking the same resource to the same agent more than once within a dialogue sequence. However, the policies we will define in Section 4 will prevent this, with obvious benefits for termination.

Example 3 *Given the three-agent agent system of example 2, ignoring any* reasons *that could be provided with dialogue moves, the agents could proceed to obtain their needed resources as follows:*

1a. `tell(`$ag1$`,`$ag2$`,(request(give(`$r2$`)) because _ ))`

2a. `tell(`$ag2$`,`$ag1$`,(request(give(`$r3$`)) because _ ))`

2b. `tell(`$ag1$`,`$ag2$`,(refuse(give(`$r3$`)) because _ ))`

1b. `tell(`$ag2$`,`$ag1$`,(accept(give(`$r2$`))))`

3a. `tell(`$ag2$`,`$ag3$`,(request(give(`$r3$`)) because _ ))`

3b. `tell(`$ag3$`,`$ag2$`,(accept(give(`$r3$`))))`

Here, $ag1$ begins with a request to obtain its needed resource $r2$ from $ag2$ (1a). This request is accepted by $ag2$ (1b) resulting in a successful dialogue instance. In between this successful dialogue instance is an unsuccessful dialogue instance (2a and 2b) between $ag2$ and $ag1$ regarding $r3$. Following the unsuccessful dialogue instance is another dialogue instance (3a and 3b) initiated by $ag2$ regarding $r3$ but this time with $ag3$. This (and hence the overall dialogue sequence) terminates successfully. At this stage, all agents have their needed resources.

3 Representation of agents

In this section we give a concrete representation of agents, that will be underpinning the generation of dialogues using the concrete policies of Section 4. This representation is given within the existing framework for Assumption-Based Argumentation (ABA) [1], briefly reviewed next, following the presentation of [4].

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 $\wp(\mathcal{L})$, where any $y \in \overline{x}$ is referred to as a *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 represent $l \leftarrow$ simply as l. As in [4], 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 $n \geq 0$.

An *argument* in favour of a sentence x in $\mathcal{L}$ supported by a set of assumptions X is a (backward) deduction from x to X, obtained by applying backwards the rules in $\mathcal{R}$. 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 *attacks* Y iff there is an argument in favour of some assumption in $\overline{x}$ supported by (a subset of) X where x is in Y. Then, a set of assumptions is deemed *admissible* iff it does not attack itself and it counter-attacks every set of assumptions attacking it. We will use the following terminology: given some $L \subseteq \mathcal{L}$, $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle \models L$ stands for "there exists a backward deduction for all sentences $l \in L$ from some admissible set of assumptions X in $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$. Computational mechanisms have been defined for computing admissible supports for conclusions (e.g. see [4]), supported by the CaSAPI (Credulous and Sceptical Argumentation Prolog Implementation) implemented system [2].

Agents in our agent systems can be represented as ABA frameworks combining all three elements of agents (their beliefs, needed and owned resources), as well as "reasoning rules" and "control information" required for supporting the policies given in Section 4.

Definition 8 *The ABA framework* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ *for an agent* $x \in A$ *has*

- $\mathcal{L} = \mathcal{L}_b \cup \mathcal{L}_p$ *such that* $\mathcal{L}_b \cap \mathcal{L}_p = \{\}$
- $\mathcal{A} = \mathcal{A}_b \cup \mathcal{A}_p$ *such that* $\mathcal{A}_b \subseteq \mathcal{L}_b$ *and* $\mathcal{A}_p \subseteq \mathcal{L}_p$
- $\mathcal{R} = \mathcal{R}_b \cup \mathcal{R}_p$ *such that* $\mathcal{R}_b \cap \mathcal{R}_p = \{\}$

Intuitively, $\mathcal{L}_b$, $\mathcal{A}_b$ with their contraries, and $\mathcal{R}_b$ allow to represent and reason with beliefs, needed and owned resources of agents: we define these componens of the ABA framework below in this section. The remaining bits are needed to support the policies of Section 4 and will be given there. Note that the overall language $\mathcal{L}$ is intended to be shared amongst agents (as the *reasons* exchanged between agents in dialogues are sets of sentences in $\mathcal{L}$).

Definition 9 (Non-assumptions in $\mathcal{L}_b$) *The non-assumptions* $\mathcal{L}_b \backslash \mathcal{A}_b$ *are all the instances of the following schemata, for* $X \in A$ *and* $R \in Res$*:*
$thisAgent(X)$ *:* the agent's own name is X
$isAgent(X)$ *:* X is an agent other than itself in the agent system
(¬) $has(X, R)$ *:* agent X has (does not have) resource R
(¬) $needs(X, R)$ *:* agent X needs (does not need) resource R

In the absence of beliefs about other agents and contrary beliefs, an agent may make assumptions about the resources agents have and need, as follows:

Definition 10 (Assumptions in $\mathcal{A}_b$ and Contraries) *The assumptions* $\mathcal{A}_b$ *(and their associated contraries) in the language* $\mathcal{L}_b$ *are all instances of the following schemata, for* $X \in A$ *and* $R \in Res$*:*
$asm(has(X, R))$ *:*an assumption that agent X has resource R*,*
with $\overline{asm(has(X, R))} = \{\neg has(X, R), has(X', R) | X' \in A, X' \neq X\}$
$asm(\neg has(X, R))$ *:* an assumption that agent X does not have resource R*,*
with $\overline{asm(\neg has(X, R))} = \{has(X, R)\}$
$asm(noAgentHas(R))$ *:* an assumption that no agent has resource R*,*
with $\overline{asm(noAgentHas(R))} = \{has(X, R) | X \in A\}$
$asm(\neg needs(X, R))$*:*an assumption that agent X does not need resource R*,*
with $\overline{asm(\neg needs(X, R))} = \{needs(X, R))\}$

Definition 11 (Initial inference rules) *The set* $\mathcal{R}_b$ *for agent* $x \in A$ *is*
$\{thisAgent(x)\} \cup \{isAgent(Y) | Y \in A, Y \neq x\} \cup$
$\{has(x, R) | R \in Res(x)\} \cup \{needs(x, R) | R \in G(x)\} \cup$
$\{\neg has(X, R) \leftarrow thisAgent(X), asm(\neg has(X, R)) | X \in A, R \in Res\} \cup$
$\{\neg needs(X, R) \leftarrow thisAgent(X), asm(\neg needs(X, R)) | X \in A, R \in Res\}$

$\mathcal{R}_b$ contains inference rules representing x's initial beliefs (first two subsets), allocated and needed resources (third and fourth subset), as well as two general-purpose inference rules (latter two sets) imposing a form of *closed world assumption* over the resources an agent does not itself have and need.

Example 4 *Given the agent system of example 1, the set of inference rules for agent $ag1$ consists initially of the* closed-world assumption *rules as well as $thisAgent(ag1)$, $isAgent(ag2)$, $has(ag1, r1)$, $needs(ag1, r2)$.*

Note that $B(x)$, as informally described in Section 2, corresponds, initially, to the first two subsets in $\mathcal{R}_b$ in definition 11. We will see, in Section 4, that, as a consequence of dialogues, agents will modify their beliefs $B(x)$ (and thus their representation as an ABA framework) by adding/removing information about resources held and/or needed by other agents and held by themselves. In the absence of any such information, by virtue of their representation as ABA frameworks, agents are allowed to make assumptions as to what resources other agents have or need, as soon as these assumptions are admissible.

4 Negotiation Policies

In this section we define two negotiation policies for agent systems with agents represented as ABA frameworks, as in Section 3: using the second such policy agents exchange reasons for their negotiation stands, thus bringing benefits to the negotiation. While defining the policies we will also provide definitions for $\mathcal{L}_p$, $\mathcal{A}_p$ and $\mathcal{R}_p$, thus obtaining the full ABA frameworks for agents. For both policies, $\mathcal{L}_p$ includes:

$committed(X, R)$: agent X is committed to obtaining resource R

$awaitingResponse(X, Y, R)$: X is awaiting a response from Y regarding R

$asm(\neg committed(X))$ $(\in \mathcal{A}_p)$: an assumption that X is not committed to obtaining anything, with $\overline{asm(\neg committed(X))} = \{committed(X, R) | R \in Res\}$

$asm(\neg awaitingResponse(X))$ $(\in \mathcal{A}_p)$: an assumption that X is not awaiting any responses, with $\overline{asm(\neg awaitingResponse(X))} = \{awaitingResponse(X, Y, R) \mid Y \in A,\ R \in Res\}$

Roughly, both policies translate as follows (but for different ways to realise GO and SA below), for a given agent x:

1. until all needed resources have been obtained repeat
   - (a) select one of the missing needed resources (say r) to obtain
   - (b) until r is obtained repeat
     - i. if *the goal of obtaining all resources is still achievable (GO)*, select a *suitable agent for asking r (SA)* (say y)

ii. if there is no such y, end in **failure**

iii. initiate dialogue with y attempting to obtain r, making any necessary updates to $\mathcal{R}$

2. end in **success**

The policies are defined as sets of *communication rules*: these have associated with them preconditions, possibly the reception of an "incoming" dialogue move, and consequences, which may be the utterance of an "outgoing" dialogue move and/or a number of revisions to the beliefs of the agent. Formally, given an agent x with $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ the ABA framework representing x, an action rule is specified as a tuple $\langle In, P, Out, X \rangle$, where In is the *incoming dialogue move*, with x as receiver, or is empty; P, the *preconditions*, is a set of sentences in the language $\mathcal{L}$; Out is the *outgoing dialogue move*, with x as utterer, or is empty; X, the *revisions*, is a set of additions (+) and deletions (-) of sentences in the language $\mathcal{L}$ to/from $B(x)$ (as represented in $\mathcal{R}_p$) and $\mathcal{R}_b$. It is intended that agent x applies a communication rule $\langle In, P, Out, X \rangle$ to generate the corresponding Out and/or X whenever the preconditions of the communication rule hold for the agent, namely In (if not empty) belongs to the set of "unprocessed" [1] dialogue moves received by the agent and $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle \models P$ holds. Namely, the computational mechanisms for ABA (see Section 3) are the main evaluation mechanisms used by agents.

4.1 Simple Negotiation Policy

For this policy, GO is always true and SA roughly corresponds to: "y has not already been asked for r in this sequence". To support this, $\mathcal{L}_p$ also includes

$cannotGive(X, R)$: agent X cannot give away resource R

$asm(canGive(X, R))$ $(\in \mathcal{A}_p)$: an assumption that agent X is in a position to give x resource R, with $\overline{asm(canGive(X, R))} = \{cannotGive(X, R)\}$

We now give the set of communication rules defining this policy: in a nutshell, agents request a resource only if they need it and agree to give away a requested resource only if they do not need it, updating their belief bases if they receive or give away a resource.

Selecting a Resource to Obtain. If the agent is not already committed to obtaining a resource, then it commits (via an addition to $\mathcal{R}_p$) to obtaining a resource which it needs and does not have:

$In_1 =$	
$P_1 =$	$\{thisAgent(X), asm(\neg committed(X)),\ needs(X, R),\ \neg has(X, R)\}$
$Out_1 =$	
$X_1 =$	$\{+committed(X, R)\}$

[1] In Section 6, footnote 2, we discuss the implementation of dialogue moves.

Sending a Request. If the agent is committed to obtaining a resource and is not already awaiting a response, then it selects an agent to request the resource from and sends the request. The selected agent is one that can be assumed to be able to give away the resource.

$In_2 =$	
$P_2 =$	$\{thisAgent(X),\ committed(X, R),\ asm(\neg awaitingResponse(X)),$ $isAgent(Y),\ asm(canGive(Y, R))\}$
$Out_2 =$	$\texttt{tell(}X\texttt{,}Y\texttt{,(request(give(}R\texttt{)) because }\{\}\texttt{))}$
$X_2 =$	$\{+awaitingResponse(X, Y, R)\}$

Note that no reasons are provided with the request. As a result of sending the request, the agent adds to $\mathcal{R}_p$ a record that it is awaiting a response.

Receiving and Responding to a Request. The agent is self-interested but generous, and so it accepts a request for a resource if it has and does not need it. Otherwise, if it does not have the resource or it needs it, the agent refuses.

$In_3 =$	$\texttt{tell(}Y\texttt{,}X\texttt{,(request(give(}R\texttt{)) because }Reasons\texttt{))}$
$P_3 =$	$\{thisAgent(X),\ has(X, R),\ \neg needs(X, R)\ \}$
$Out_3 =$	$\texttt{tell(}X\texttt{,}Y\texttt{,(accept(give(}R\texttt{))))}$
$X_3 =$	$\{-has(X, R)\}$

$In_4 =$	$\texttt{tell(}Y\texttt{,}X\texttt{,(request(give(}R\texttt{)) because }Reasons\texttt{))}$
$P_4 =$	$\{thisAgent(X),\ \neg has(X, R)\ \}$
$Out_4 =$	$\texttt{tell(}X\texttt{,}Y\texttt{,(refuse(give(}R\texttt{)) because }\{\}\texttt{))}$
$X_4 =$	$\{\}$

$In_5 =$	$\texttt{tell(}Y\texttt{,}X\texttt{,(request(give(}R\texttt{)) because }Reasons\texttt{))}$
$P_5 =$	$\{thisAgent(X),\ needs(X, R)\ \}$
$Out_5 =$	$\texttt{tell(}X\texttt{,}Y\texttt{,(refuse(give(}R\texttt{)) because }\{\}\texttt{))}$
$X_5 =$	$\{\}$

In sending acceptance, the agent updates its belief base (represented by a subset of $\mathcal{R}_b$) so that it no longer has the resource just given away. No reasons are provided with the refusals. Reasons in the request, if any, are ignored.

Receiving a Response. The agent processes acceptance and refusals as follows:

$In_6 =$	$\texttt{tell(}Y\texttt{,}X\texttt{,(accept(give(}R\texttt{))))}$
$P_6 =$	$\{thisAgent(X),\ awaitingResponse(X, Y, R)\ \}$
$Out_6 =$	
$X_6 =$	$\{+has(X, R), -awaitingResponse(X, Y, R), -committed(X, R)\}$

$In_7 =$	$\texttt{tell(}Y\texttt{,}X\texttt{,(refuse(give(}R\texttt{)) because }Reasons\texttt{))}$
$P_7 =$	$\{thisAgent(X),\ awaitingResponse(X, Y, R)\ \}$
$Out_7 =$	
$X_7 =$	$\{+cannotGive(Y, R), -awaitingResponse(X, Y, R)\}$

In receiving acceptance, the agent updates its $\mathcal{R}_b$ and $\mathcal{R}_p$ to record that it now has the requested resource ($\mathcal{R}_b$) and removes its commitment to obtaining it and to wait for a response about it ($\mathcal{R}_p$). Then, the agent is free to obtain any other missing needed resources. In receiving refusal, on the other hand, the agent adds to $\mathcal{R}_p$ that the refusing agent cannot give away the resource. Then, the agent will proceed to ask another agent that it has not already asked, if any, for the missing needed resource.

4.1.1 Demonstrating the simple negotiation policy

Example 5 *Let* $A = \{ag1,\ ag2,\ ag3\}$ *be an agent system as follows:* $G(ag1) = \{r2,\ r3\}$, $R(ag1) = \{\}$, $r2 \in R(ag2)$, $r2 \notin G(ag2)$, $r3 \in R(ag3)$, $r3 \notin G(ag3)$. *We focus on* $ag1$ *and the dialogues it initiates in pursuit of its needed resources* $r2$ *and* $r3$. *We assume* $ag1$ *begins with no beliefs other than knowing the agents in the system and the resources it itself has and needs, i.e.* $B(ag1) = \{thisAgent(ag1),\ isAgent(ag2),\ isAgent(ag3),\ needs(ag1, r2),\ needs(ag1, r3)\}$.

Initially, $ag1$ *can commit to obtaining* $r2$ *or* $r3$ *according to precondition* P_1. *Assume it commits to obtaining* $r2$ *first, thus it adds* $committed(ag1, r2)$ *to its* $\mathcal{R}_p$ *according to* X_1. *Next it can send a request for* $r2$ *to* $ag2$ *or* $ag3$ *according to* P_2. *Assume it selects* $ag2$*, then it utters* `tell(`$ag1$`,`$ag2$`,(request(give(`$r2$`)) because` $\{\}$`))` *according to* Out_2 *and adds* $awaitingResponse(ag1, ag2, r2)$ *to its* $\mathcal{R}_p$ *according to* X_2. *At this point, neither* P_1 *nor* P_2 *hold for* $ag1$*, so* $ag1$ *simply waits for a response from* $ag2$.

$ag2$ *receives the request and accepts because* P_3 *holds. Neither* P_4 *nor* P_5 *hold.* $ag2$ *utters* `tell(`$ag2$`,`$ag1$`,(accept(give(`$r2$`))))` *according to* Out_3*, modifying its belief base (in* $\mathcal{R}_b$*) according to* X_3.

$ag1$ *receives the acceptance (*In_6*) and since* P_6 *holds, updates its* $\mathcal{R}_b \cup \mathcal{R}_p$ *according to* X_6*, removing its commitment to obtaining* $r2$ *which it now has. Assume next* $ag1$ *commits to obtaining* $r3$ *and selects* $ag2$ *to request* $r3$ *from, as it did for* $r2$*, then it makes an utterance according to* Out_2.

$ag2$*, however, this time round, refuses because* P_4 *holds.* $ag2$ *makes an utterance* `tell(`$ag2$`,`$ag1$`,(refuse(give(`$r3$`)) because` $\{\}$`))` *according to* Out_4.

$ag1$ *receives the refusal (*In_7*) and since* P_7 *holds, performs updates to its* $\mathcal{R}_p$ *according to* X_7*, in particular adding the belief* $cannotGive(ag2, r3)$. *Subsequently,* $ag1$ *requests and successfully obtains* $r3$ *from* $ag3$ *as it did* $r2$ *from* $ag2$. *At this point,* $ag1$ *does not initiate any more dialogues since it has obtained all its needed resources and hence neither* P_1 *nor* P_2 *hold hereafter.*

Example 6 *Assume the exact same setup as in example 5 except that* $r2 \in G(ag2)$*, i.e.* $ag2$ *has and needs* $r2$*, and* A *includes several other agents. Now, when* $ag1$ *requests* $r2$ *from* $ag2$*,* $ag2$ *refuses since it needs* $r2$ *(*P_5 *holds rather than* P_3 *or* P_4*). Following this, after processing the refusal according to* P_7/X_7*,* $ag1$ *requests* $r2$ *from* $ag3$*, which refuses because it does not have* $r2$ *(according to* P_4*).* $ag1$ *would continue requesting* $r2$ *from all other agents in the system, despite having already asked* $ag2$ *which has* $r2$. *This is wasteful as resources are single-unit and if* $ag1$ *knew that* $ag2$ *has* $r2$*, it could stop asking.*

4.2 Reason-based Negotiation Policy

As demonstrated in example 6, because no reasons are provided with refusals, the agent receiving a refusal cannot ascertain as to why its request has been refused. Thus, for the sake of completeness, the agent must proceed to ask every agent in the system until an agent accepts its request or until all agents have been asked. This drawback can be eliminated by providing reasons for the refusal. These reasons can then be used in subsequent dialogues or to terminate the sequence. For the reason-based policy, GO roughly corresponds to: "no other agent is known to have and need r" and SA roughly corresponds to: "y can be assumed to have and not need r". To support this, $\mathcal{L}_p$ given at the beginning of Section 4 needs to be extended to also include:

$goalUnachievable(X)$: *agent X's goal is unachievable*

$asm(goalAchievable(X))$ $(\in \mathcal{A}_p)$: *an assumption that agent X's goal is achievable, with* $\overline{asm(goalAchievable(X))} = \{goalUnachievable(X)\}$

where $\mathcal{R}_p$ includes all instances (for $X \in A$ and $R \in Res$) of the schema:

$$goalUnachievable(X) \leftarrow needs(X,R), has(Y,R), needs(Y,R), Y \neq X (1)$$

Indeed, since we work in a single-unit resource setting and the goal of each agent is to obtain all of its needed resources, if an agent x believes another agent to have and need a resource that x itself needs, then x will be aware that it will not be able to obtain all the resources it needs.

Selecting a Resource to Obtain. This is as for the simple policy of Section 4.1.

Sending a Request. As for the simple policy, the agent selects an agent to request the needed resource from if it is not already awaiting a response. However, the agent selected now is one that can be assumed to have and not need the resource. Also, the request is only sent if the agent can assume its goal to be achievable.

$In_{1r} =$	
$P_{1r} =$	$\{thisAgent(X),\ committed(X,R),\ asm(\neg awaitingResponse(X))$ $isAgent(Y),\ asm(has(Y,R)),\ asm(\neg needs(Y,R)),$ $asm(goalAchievable(X))\}$
$Out_{1r} =$	`tell(`X`,`Y`,(request(give(`R`)) because` $\{needs(X,R), \neg has(X,R)\}$`))`
$X_{1r} =$	$\{+awaitingResponse(X,Y,R)\}$

Note that the requesting agent now provides reasons with the request.

Receiving and Responding to a Request.

In_{2r} =	$\texttt{tell(}Y\texttt{,}X\texttt{,(request(give(}R\texttt{)) because }\{\mathit{needs}(Y,R), \neg \mathit{has}(Y,R)\}\texttt{))}$
P_{2r} =	$\{\mathit{thisAgent}(X),\ \mathit{has}(X,R),\ \neg \mathit{needs}(X,R)\ \}$
Out_{2r} =	$\texttt{tell(}X\texttt{,}Y\texttt{,(accept(give(}R\texttt{))))}$
X_{2r} =	$\{+\mathit{needs}(Y,R),\ +\mathit{has}(Y,R),\ -\neg \mathit{has}(Y,R),\ -\mathit{has}(X,R)\}$

The agent accepts the request, as in the corresponding rule for the simple policy, but updating its belief base ($\mathcal{R}_b$) in a richer manner (using the reasons provided).

In_{3r} =	$\texttt{tell(}Y\texttt{,}X\texttt{,(request(give(}R\texttt{)) because }\{\mathit{needs}(Y,R), \neg \mathit{has}(Y,R)\}\texttt{))}$
P_{3r} =	$\{\mathit{thisAgent}(X),\ \mathit{has}(X,R),\ \mathit{needs}(X,R)\ \}$
Out_{3r} =	$\texttt{tell(}X\texttt{,}Y\texttt{,(refuse(give(}R\texttt{)) because }\{\mathit{has}(X,R), \mathit{needs}(X,R)\}\texttt{))}$
X_{3r} =	$\{+\mathit{needs}(Y,R),\ +\neg \mathit{has}(Y,R),\ -\mathit{has}(Y,R)\}$

In_{4r} =	$\texttt{tell(}Y\texttt{,}X\texttt{,(request(give(}R\texttt{)) because }\{\mathit{needs}(Y,R), \neg \mathit{has}(Y,R)\}\texttt{))}$
P_{4r} =	$\{\mathit{thisAgent}(X),\ \neg \mathit{has}(X,R),\ \mathit{has}(Z,R),\ \mathit{needs}(Z,R), Z \neq Y, Z \neq X\ \}$
Out_{4r} =	$\texttt{tell(}X\texttt{,}Y\texttt{,(refuse(give(}R\texttt{))}$ $\texttt{because }\{\neg \mathit{has}(X,R), \mathit{has}(Z,R), \mathit{needs}(Z,R)\}\texttt{))}$
X_{4r} =	$\{+\mathit{needs}(Y,R),\ +\neg \mathit{has}(Y,R),\ -\mathit{has}(Y,R)\}$

In_{5r} =	$\texttt{tell(}Y\texttt{,}X\texttt{,(request(give(}R\texttt{)) because }\{\mathit{needs}(Y,R), \neg \mathit{has}(Y,R)\}\texttt{))}$
P_{5r} =	$\{\mathit{asm}(\mathit{noAgentHas}(R))\}$
Out_{5r} =	$\texttt{tell(}X\texttt{,}Y\texttt{,(refuse(give(}R\texttt{)) because }\{\mathit{asm}(\mathit{noAgentHas}(R))\}\texttt{))}$
X_{5r} =	$\{+\mathit{needs}(Y,R),\ +\neg \mathit{has}(Y,R),\ -\mathit{has}(Y,R)\}$

The agent refuses the request, as in the corresponding (two) rules for the simple policy, but providing here the appropriate reasons. Note that, for any request received at any point in time, one and only one of P_{2r}-P_{5r} is guaranteed to hold, since P_{2r}-P_{5r} are exhaustive and mutually exclusive. Thus, the agent generates one and only one *final dialogue move* for every *initial dialogue move* received.

Receiving a Response. The agent updates its set of inference rules ($\mathcal{R}_b \cup \mathcal{R}_p$) upon receiving acceptance or refusal, taking the provided reasons into account.

In_{6r} =	$\texttt{tell(}Y\texttt{,}X\texttt{,(accept(give(}R\texttt{))))}$
P_{6r} =	$\{\mathit{thisAgent}(X),\ \mathit{awaitingResponse}(X,Y,R)\ \}$
Out_{6r} =	
X_{6r} =	$\{+\mathit{has}(X,R),\ +\neg \mathit{has}(Y,R),\ -\neg \mathit{has}(X,R),\ -\mathit{has}(Y,R),$ $-\mathit{awaitingResponse}(X,Y,R), -\mathit{committed}(X,R)\}$

In_{7r} =	$\texttt{tell(}Y\texttt{,}X\texttt{,(refuse(give(}R\texttt{)) because }\{\mathit{has}(Y,R), \mathit{needs}(Y,R)\}\texttt{))}$
P_{7r} =	$\{\mathit{thisAgent}(X),\ \mathit{awaitingResponse}(X,Y,R)\ \}$
Out_{7r} =	
X_{7r} =	$\{+\mathit{has}(Y,R),\ +\mathit{needs}(Y,R),\ -\neg \mathit{has}(Y,R),$ $-\mathit{awaitingResponse}(X,Y,R)\}$

In_{8r} =	`tell(`Y`,`X`,(refuse(give(`R`))` `because` $\{\neg has(Y,R), has(Z,R), needs(Z,R)\}$`))`
P_{8r} =	$\{thisAgent(X),\ awaitingResponse(X,Y,R)\ \}$
Out_{8r} =	
X_{8r} =	$\{+\neg has(Y,R),\ +has(Z,R),\ +needs(Z,R),$ $-has(Y,R),\ -\neg has(Z,R),\ -awaitingResponse(X,Y,R)\}$

In_{9r} =	`tell(`Y`,`X`,(refuse(give(`R`))` `because` $\{asm(noAgentHas(R))\}$`))`
P_{9r} =	$\{thisAgent(X),\ awaitingResponse(X,Y,R)\ \}$
Out_{9r} =	
X_{9r} =	$\{+\neg has(Y,R),\ -has(Y,R),\ -awaitingResponse(X,Y,R)\}$

4.2.1 Demonstrating the reason-based negotiation policy

Example 7 *Assume an agent system as given in example 6. Assume also that* $ag1$ *commits to obtaining* $r2$ *according to* P_1/X_1 *and* $ag1$ *selects* $ag2$ *to request* $r2$ *from according to* P_{1r}/X_{1r}*. Thus* $ag1$ *makes the utterance* `tell(`$ag1$`,` $ag2$`,` `(request(give(`$r2$`))` `because` $\{needs(ag1,r2),\ \neg has(ag1,r2)\}$`))` *according to* Out_{1r}*, providing reasons for its request.*

$ag2$ *refuses according to* P_{3r} *and makes the utterance* `tell(`$ag2$`,` $ag1$`,` `(refuse(give(`$r2$`))` `because` $\{has(ag2,r2), needs(ag2,r2)\}$`))` *according to* Out_{3r}*, providing reasons for its refusal.*

$ag1$ *receives the refusal according to* In_{7r} *and updates its belief base according to* X_{7r}*, making use of the reasons in the refusal. In this case, differently from example 6 where the simple policy was used,* $ag1$ *does not unnecessarily request* $r2$ *from any other agent since the precondition* P_{1r} *for initiating requests will no longer hold.* $ag1$ *ends in failure, correctly, at this point.*

5 Properties of the Policies

The agent policies have been defined bearing certain properties in mind, as discussed in this section. Informal proofs can be found in Appendices A and B.

- *Consistency of beliefs.* The revisions associated with the agent's communication rules are such that (i) it never believes an agent to both have and not have a particular resource, and (ii) it never believes any resource to be held by two agents simultaneously. Hence the agent's belief base remains consistent after any belief update.

- *Termination of dialogue instances.* Dialogue instances generated by deploying the policies are guaranteed to terminate since the conditions for responding to a *dialogue initial move* (i.e. a request) are exhaustive (namely, one will always hold). Hence a *dialogue final move* (i.e. an acceptance or refusal) is always generated and sent for every *dialogue initial move* received.

- *Non-duplicate requests.* The preconditions and belief updates associated with the communication rules are such that agents do not request the same resource from the same agent more than once.
- *Termination of dialogue sequences.* Since (i) each dialogue instance terminates, (ii) no agent is asked twice for the same resource, (iii) there are only a finite number of agents, and (iv) agents need only a finite number of resources, no agent will continue requesting resources forever and the dialogue sequences generated by deploying either policy are finite.
- *Completeness.* For an agent system for which a solution for the r.r.p. exists, each agent will get the resources it needs and hence the solution will be found.

In addition, the reason-based policy has been defined with “efficiency” in mind. We measure efficiency of the reason-based policy in comparison with the simple policy. Concretely, (i) dialogue sequences using the reason-based policy consist of no more dialogue instances than those using the simple policy, with no compromise in the quality of solutions. This implies completeness, in the same sense as for the simple policy. Also, (ii) the agents that need to be contacted during a dialogue sequence using the reason-based policy is the same as or a subset of those that need to be contacted using the simple policy.

6 Conclusions

We have presentented a framework whereby agents can negotiate the reallocation of resources (needed to achieve their goals) by using one of two policies. These policies allow agents to generate negotiation dialogues, and are empowered by an argumentation framework for representing and reasoning with the agents’ minds (their beliefs, the resources they own and need) and for evaluating the preconditions of the communication rules composing the policies. The two policies differ in the amount of information that agents exchange using them: with the second, reason-based policy agents provide reasons for their requests and refusals of requests; with the first, simple policy they do not provide these reasons. These reasons can be seen as simple arguments to explain requests/refusals. We have seen that the second, reason-based policy is more “effective” for agents to obtain all resources they need or realise that they cannot obtain (all of) them. Future work will look at building upon the realisation that a plan is unachievable by allowing agents to formulate alternative plans rather than ending in failure.

We have implemented our framework using Jade (Java Agent DEvelopment Framework) [6] and CaSAPI (see Section 3) interfacing the two by means of PrologBeans [8]. We make use of Jade’s naming, messaging [2] and yellow page services, as well as Jade’s capability to support the scheduling of cooperative

[2]Each Jade agent comes with an associated message queue for storing incoming messages. As messages are processed, they are removed from the message queue.

behaviours, to represent the agents and their associated communication rules. We use CaSAPI to represent the belief bases of the agents and to evaluate the preconditions of the communication rules. Experimental results can be found in Appendix C.

We have made a number of simplifying assumptions. Although our policies are generic, and could in principle be used by any agents, we have assumed that our agents use ABA for decision-making and reasoning: this is not suitable for supporting open multi-agent systems where negotiation is likely to be required. Both policies are designed so that agents are truthful and only provide reasons that they hold as true: again, it will not be easy to enforce this in open systems. The use of argumentation is limited to internal decision-making and reasoning and exchange of simple reasons: in general, it would be useful to allow agents to argue with one another, for example for persuasion or in case agents are found to be untruthful. Agents are assumed to have fixed goals (the set of resources they need): in general, agents may have a number of conflicting goals (possibly with preferences) and alternative plans to achieve different goals (each requiring different sets of resources). Resources are assumed to be single unit, indivisible and unsharable: this is often not the case in applications. Future work will look at removing these assumptions.

Acknowledgements

The second author has been supported 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] CaSAPI. http://www.doc.ic.ac.uk/∼dg00/casapi.html.

[3] Pieter Dijkstra, Henry Prakken, and Kees de Vey Mestdagh. An implementation of norm-based agent negotiation. In *ICAIL 07*, pages 167–175, 2007.

[4] Dorian Gaertner and Francesca Toni. Hybrid argumentation and its properties. In Antony Hunter, editor, *COMMA 08*. IOS Press, 2008.

[5] Adil Hussain and Francesca Toni. On the benefits of argumentation for negotiation - preliminary version. In *EUMAS*, 2008.

[6] Jade. http://jade.tilab.com.

[7] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[8] PrologBeans. http://www.sics.se/sicstus/docs/latest3/html/prologbeans/.

[9] Fariba Sadri, Francesca Toni, and Paolo Torroni. Dialogues for negotiation: Agent varieties and dialogue sequences. In *ATAL*, pages 405–421, 2001.

[10] Jelle van Veenen and Henry Prakken. A protocol for arguing about rejections in negotiation. In *ARGMAS*, pages 138–153, 2005.

A Properties of the Simple Policy

We discuss informally some properties for an agent system whose agents use the simple negotiation policy described in Section 4.

- *Termination of dialogue instances.* Dialogue instances generated by deploying the policy are guaranteed to terminate since the conditions P_3-P_5 for responding to a *dialogue initial move* (i.e. a request) are exhaustive (i.e. either an agent has and does not need a requested resource, or it does not have it or it needs it). Hence a *dialogue final move* (i.e. an acceptance or refusal) is always generated and sent for every *dialogue initial move* received.
- *Non-duplicate requests.* If an agent x sends a request to an agent y for a resource r, the response is either acceptance or refusal:
  - in the case of acceptance, a belief $has(x, r)$ is added to $\mathcal{R}_b$ of x and commitment to obtain r is removed. Subsequently, because of the way P_1 is defined and since x will never give away r, a commitment to obtain r will never be added again and thus x will not request r from y again.
  - in the case of refusal, a record $cannotGive(y, r)$ is added to $\mathcal{R}_p$ of x. Subsequently, because this record is never removed and because of the way P_2 is defined, x will never again request r from y.
- *Termination of dialogue sequences.* Since (i) each dialogue instance terminates, (ii) no agent is asked twice for the same resource, (iii) there are only a finite number of agents, and (iv) agents need only a finite number of resources, no agent will continue requesting resources forever and the dialogue sequences generated by deploying this policy are finite.
- *Completeness.* An agent x attempting to obtain a resource r continues requesting r from the other agents in the system one by one until:

  - *An agent that has r and does not need it is asked:* in such a case, since agents give away resources that they do not need (P_3), r will be obtained by x and the dialogue sequence will terminate successfully, as expected.
  - *All agents have been asked:* This is because no agent has r or because an agent has r but needs it. Either way, the dialogue sequence will terminate unsuccessfully, as expected, since no more requests will be initiated by x and the last dialogue instance in the sequence was an unsuccessful one.

Taking this a step further, for an agent system for which a solution for the r.r.p. exists, the solution will be found, since each agent will get the resources it needs.

B Properties of the Reason-Based Policy

We discuss informally some properties for a system of agents whose agents use the reason-based policy described in Section 4.

- *Consistency of Beliefs.* An agent's belief base (represented by a subset of $\mathcal{R}_b$) remains consistent after any belief update since (i) it never believes an agent to both have and not have a particular resource, and (ii) it never believes any resource to be held by two agents simultaneously. The proof of this is roughly as follows:

  (i) Whenever a belief of the form $has(Ag, R)$ is added to $\mathcal{R}_b$, a belief $\neg has(Ag, R)$ is removed, and vice versa.

  (ii) Looking at the cases where a belief of the form $has(\_, R)$ is added to $\mathcal{R}_b$:
    - in the case of X_{2r} and X_{6r}, such a belief is added but one is also removed.
    - in the case of X_{7r} and X_{8r}, such a belief is added but none are removed. This is not problematic since X_{7r} and X_{8r} only occur in response to a request, and an agent only requests a resource if it does not believe any agent to have that resource.

- *Mutual Exclusion of Responding Conditions* We show here that the responding conditions P_{2r}-P_{5r} of the argumentative negotiation policy are mutually exclusive. We demonstrate this for an agent *ag1* that has been requested to give away r, as follows:
  - *If P_{2r} holds*, then *ag1* believes *has(ag1,r)* and thus P_{4r} and P_{5r} cannot hold because the necessary assumptions cannot be made. Also, *ag1* can assume *asm(¬needs(ag1,r))* thus it cannot simultaneously believe *needs(ag1,r)* and thus P_{3r} cannot also hold.

  - *If P_{3r} holds*, then *ag1* believes *has(ag1,r)* and thus P_{4r} and P_{5r} cannot hold because the necessary assumptions cannot be made. Also, *ag1* believes *needs(ag1,r)* thus it cannot simultaneously assume *asm(¬needs(ag1,r))* and thus P_{2r} cannot also hold.
  - *If P_{4r} holds*, then *ag1* can assume *asm(¬has(ag1,r))* and thus P_{2r} and P_{3r} cannot hold because *ag1* cannot also believe *has(ag1,r)*. Also, *ag1* believes *has(Ag,r)* for some ground *Ag*, thus *ag1* cannot simultaneously assume *asm(noAgentHas(r))* and thus P_{5r} cannot also hold.
  - *If P_{5r} holds*, then *ag1* can assume *asm(noAgentHas(ag1,r))* and thus P_{2r}, P_{3r} and P_{4r} cannot hold because *ag1* cannot also believe *has(Ag,r)* for any *Ag*.

- *Exhaustiveness of Responding Conditions* We show here that the responding conditions P_{2r}-P_{5r} of the argumentative negotiation policy are exhaustive, as follows: P_{2r} and P_{3r} are candidates for holding if an agent (say *ag1*) has the requested resource r (i.e. *ag1* believes *has(ag1,r)*). In such a case, since *ag1* assumes *asm(¬needs(ag1,r))* if and only if it does not believe *needs(ag1,r)*, one of P_{2r} or P_{3r} is guaranteed to hold. If, on the other hand, *ag1* does not have the requested resource r and has no belief as to which agent has r, then P_{5r} holds. If *ag1* does not have the requested resource r but does have some belief as to what agent has r, then P_{4r} holds. This covers all eventualities and hence one of P_{2r}-P_{5r} is guaranteed to hold.

- *Termination of dialogue instances.* Dialogue instances are guaranteed to terminate since the preconditions P_{2r}-P_{5r} are exhaustive as discussed earlier.

- *Non-duplicate requests.* Having sent a request to an agent (say y) for a resource (say r), the agent (say x) receives one of four responses:
  - *Acceptance:* x updates its belief base adding the belief that it has r. Subsequently, x does not request r from any agent, let alone y.
  - *Refusal because y has and needs r:* x updates its belief base accordingly. As a result, x does not initiate any more requests for any resource from any agent since, by rule (1) in $\mathcal{R}_p$, it believes its goal to be unachievable.
  - *Refusal because another agent (z) is believed to have and need r:* x updates its belief base accordingly. As a result, x does not initiate any more requests for any resource from any agent since, by rule (1) in $\mathcal{R}_p$, it believes its goal to be unachievable.
  - *Refusal because y assumes no agent to have r:* x updates its belief base to include $\neg has(y, r)$. As a result, x will not select y to request r from. The belief $\neg has(y, r)$ is only removed if x is subsequently informed by another agent that y has and needs r, in which case x

would not initiate any more requests for any resource from any agent since, by rule (1) in $\mathcal{R}_p$, it would believe its goal to be unachievable.

- *Termination of dialogue sequences.* Similarly as for the simple policy.
- *Efficiency and Completeness.* A sketch of the proof of efficiency and completeness as described in Section 5 is as follows: An agent x attempting to obtain a resource r, continues requesting r from the other agents in the system one by one (without repeating any request) until:
  - *All agents have been asked and no agent has r:* In such a case, the number of dialogue instances and agents that will be contacted using either policy is the same.
  - *An agent y, which has r and does not need it, is asked:* Using either policy, y accepts the request and x does not subsequently request r from any other agent. Thus, the number of dialogue instances and agents that will be contacted using either policy is the same.
  - *An agent y, which has r and needs it, is asked:* Using either policy, y refuses the request. However, in the case of the reason-based policy, y provides additional information and x will not subsequently request r from any other agent. In the case of the simple policy, on the other hand, x will continue unsuccessfully requesting r from all the remaining agents.
  - *An agent y, which believes z to have and need r, is asked:* Using either policy, y refuses the request. However, in the case of the reason-based policy, y provides additional information and x will not subsequently request r from z or any other agent (since we are assuming that agents are truthful). In the case of the simple policy, on the other hand, x will continue unsuccessfully requesting r from all the remaining agents.

  In addition, if an agent (say y) requests a resource (say r) from another agent (say x), providing the reason that it needs r, then x will not subsequently request r from y, resulting in a further reduction in the number of dialogue instances instantiated and agents contacted by x.

C Experimental Results

We show in Tables 1 and 2 some experimental results obtained from the implementation. The tables show how the total number of requests instantiated in an agent system varies depending on the policy of the agents – simple or reason-based. In all tests, agents begin with no information about the other agents except to know which other agents make up the agent system. We vary the number of agents in the agent system in turn and compare the total number of requests instantiated by all agents.

	Number of Agents				
Agent Type	2	3	4	5	10
Non-Argumentative	2	6	12	20	90
Argumentative	1	3	6	10	45

Table 1: Total number of requests instantiated in an agent system A wherein $\forall x \in A$, $G(x) = \{r\}$ and $r \notin Res(x)$, i.e. agents need one resource only, r, which none of them has.

	Number of Agents				
Agent Type	2	3	4	5	10
Non-Argumentative	1	4	9	16	81
Argumentative	1	2-3	3-6	4-10	9-45

Table 2: Total number of requests instantiated in an agent system as before, but in which one (and only one) agent has the needed resource r, i.e. $\exists x \in A$, $G(x) = \{r\}$ and $r \in Res(x)$. Note that the number of requests displayed is not a fixed number but rather a range of numbers. This is because the total number of requests instantiated depends on the order in which agents request the needed resource from the other agents. Thus we ran the experiment a number of times and provide the upper and lower limits.