Liberalizing Protocols for Argumentation in Multi-agent Systems

Gerard A.W. Vreeswijk

Dept. of Computer Science
Utrecht University, The Netherlands
gv@cs.uu.nl

Abstract. This paper proposes a liberalized version of existing truth-finding protocols for argumentation, such as the standard two-agent immediate-response protocol for computing the credulous acceptance of conclusions in an argument system. In the new setup agents decide autonomously which issues need to be discussed, when to query other agents, when to keep on querying other agents, and when to settle for an answer. In this way, inter-agent disputes are regulated by the agents themselves, rather than by following an outlined protocol. The paper concludes with a prototype implementation and with a comparison of related work on conversation analysis and computational dialectic.[1]

1 Introduction

Argumentation has become increasingly important in multi-agent system (MAS) research. Modern MAS models require that agents are able to argue, for example to support their position in a negotiation or to explain a possibly controversial decision.

A great deal of research on defeasible reasoning and formal argumentation has been done in the past few years, and also a great deal of research on inter-agent inquiry dialogue has been accomplished. However, most research on argumentation in AI is devoted to monological (single-agent) algorithms and dialogical two-party immediate response dialectics that are sound and complete with respect to a particular argument semantics. Examples of such semantics are the grounded extension semantics, the stable extension semantics and the preferred extension semantics [8, 32]. Research on inter-agent inquiry, on the other hand, is concerned with studying sequences of conversation at the speech act level that are useful, orderly, effective [11, 25, 28] and sufficiently controllable by the agents that use them [2].

A remarkable difference between the two approaches is that argumentation dialogues are often extremely constrained and deterministic, while inter-agent inquiry dialogues are less constrained but also less concerned with getting the underlying argument semantics right [31]. Recently, a number of proposals have

[1] A (colorful and instructive) poster based on a shorter version of this paper was presented at AAMAS'05 [34]. The poster itself can be viewed at http://www.cs.uu.nl/~gv/abstracts/liberal_protocol_poster.pdf.

S. Parsons et al. (Eds.): ArgMAS 2005, LNAI 4049, pp. 182–198, 2006.

been made to connect the two approaches, for example by dropping protocol constraints [23] or, conversely, by formulating desiderata for argumentation protocols [15, 17].

This paper proposes a minimalistic but complete model for inter-agent argumentation that is less constrained than existing argumentation protocols. The model is minimalistic in the sense that the agent architecture, the internal knowledge representational language and the message format are minimalistic and contain just enough detail to "keep the agents going". The model is complete in the sense that it describes the entire setup—from agent internals to communication language—and possesses enough detail to obtain a runnable MAS.

The purpose of the model is give the minimal means with which agents can engage in a dispute that is brought about by the agents themselves (autonomous agent perspective) rather than that the agents follow a fixed and external protocol (defeasible argumentation perspective). The resulting system is suitable for parametrization, experimentation and analysis.

The paper is structured as follows. In Sec. 2 the global setup is described. Sec. 3 describes the agents architecture, and Sec. 4 describes agents actions in more detail. The paper concludes with a discussion of a prototype implementation and with a comparison of related work on conversation analysis and computational dialectic.

2 Global Setup

The global setup consists of a set $\mathcal{A} = \{A_1, \ldots, A_n\}$ of agents ($n \geq 2$) and a public communication medium T, called *the table*. T can be seen as a blackboard, or as "open air," by means of which agents are able to exchange messages in public. More specifically, T is a passive object with two essential methods, viz.

```
put(m: message)
get(t1: time, t2: time): setofMessages
```

By way of the second method agents can retrieve all messages that were uttered between time points t_1 and t_2.

Experiments are performed in runs. A run is a complete session in which agents are initialized by the programmer, and then exchange messages autonomously until no agent activity is observed within some fixed time period. At the start of each run each agent receives a number of propositions from the programmer to fill its belief base with. The initial goal base of each agent cannot be programmed and consists of one action, viz. `listen`. A typical run starts with one or more agents that have a computed interest in determining the credibility of one or more propositions. These propositions are put on the table in the form of queries. These queries invoke a dispute. This dispute ends as soon as all agents have lost all incentives to utter speech acts (typically queries and answers to queries).

For the sake of simplicity the present setup assumes that agents comply to specific (and admittedly often unrealistic) maxims of co-operation. In particular, it is assumed that agents are honest and credulous. Honesty corresponds to the

Gricean maxim that agents are forbidden to put forward information they do not believe; credulousness corresponds to the property that agents believe what they are told. I do not think that it is difficult to extend the present setup to a scenario where these constraints are dropped. Evidently this avenue goes beyond the scope of this paper.

For the same reasons of simplicity, the model does not assume that agents are perfect reasoners or communicators. In particular an agent can be programmed such that it prioritizes communication at the expense of logical inference. Conversely, it is possible to program "ponderers" that prioritize internal inference at the cost of communication. Obviously, both extremes are undesirable and the programmer is responsible for achieving the right balance. The programmer can achieve this balance by ensuring that (most) communication actions invoke logical actions and conversely (which is a natural phenomenon).

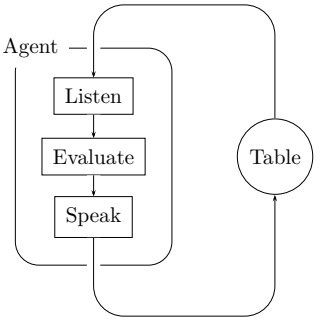

Fig. 1. Serial deliberation

3 Agent Architecture

An agent $A = (B, P, G)$ is a daemon that possesses a declarative belief base B, a procedural belief base P, and a goal base, or agenda, G.

- The declarative belief base B contains propositions about the state of the world, formulated in a simple logical object language, annotated with information that pertains to the proposition's origin, the proposition's degree of belief, and other attributes.
- The procedural belief base P contains information that is concerned with internal procedural matters. An example of a procedural item is a (private) method that returns a pointer to the next unread message (a so-called *bookmark*).
- The goal base G is a (private) priority queue filled with actions. Actions can either be internally or externally directed. Thus, agents can schedule belief updates as well as sending messages.

Agents are not able to inspect, modify, or communicate about the contents of P. Therefore, the objective in designing A is to put as much as possible of P in B and G, so that agents can reason and communicate about their beliefs.

3.1 Deliberation Cycle

Each agents runs an eternal loop, also called a *cycle* [3], or a *deliberation cycle* [6, 12]. Contrary to first-generation deliberation cycles this loop is not serial

(Fig. 1) but prioritized (Fig. 2).[2] At every pass of the loop the agent takes an action from the priority queue and executes it. A typical execution of an action amounts to doing a few operations in the internal representation format of the agent, interspersed with (or followed by) scheduling some new actions. The priority of these new actions depends on their type and the priority and contents of the action that caused the scheduling of the new actions.

Actions are not executed in the order in which they are put on the agenda, but according to they priority. If a set of actions nevertheless must be executed in succession, this can be accomplished in two ways.
(1) The first way is to simply concatenate the actions as a plan in the body of the action statement a. In this way, all actions in the body of a are executed immediately if the head of a is taken from the priority queue.

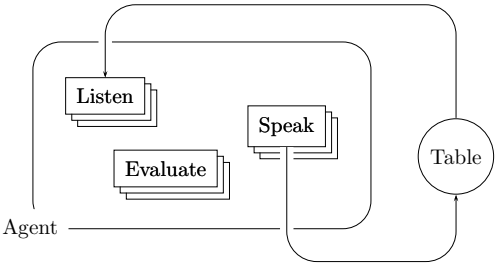

Fig. 2. Prioritized deliberation

(2) The second way is to assign decreasing priorities but equal activation factors to a list of actions. In this way the actions are guaranteed to be executed in succession, be it that they are likely to be interleaved with other actions. For many types of actions this is no problem.

An example of a typical action is "Listen" (Action 1). When this action is taken from the queue, the agent fetches the last unread message from the table. If this action succeeds, and the message is not from the agent itself and not addressed to another agent, the agent schedules a "process-message" activity. Independently a next listen activity is scheduled.

3.2 Goal Base

An agent's goal base, or *agenda*, is a priority queue filled with actions. To ensure that all actions are eventually executed, this priority queue is equipped with a scheduler that is derived from the standard priority schedulers used in operating systems theory [27].

The scheduler works as follows. Contrary to [3, 6, 12], actions do not have preconditions but possess, besides an action-inherent priority, a so-called *activation factor*. (If one of them is missing, a reasonable default is used.) When the agenda is initialized, the agenda is given its own activation factor as well. The activation factor of an agenda represents the nominal speed with which scheduled actions rise ("bubble") to the top of the priority queue, once they are put on the agenda. Thus, agendas as well as prioritized actions possess an activation factor.

Each time an agent puts a new action a on its agenda, the priority of a is increased with a's activation factor times the activation factor of the agenda. The

[2] Cf. relation work in Section 7.

Action 1. Listen

```
 1: table.get( procedural.first-unread-message )
 2: if defined message then
 3:    if message.from == my-own-name then
 4:       Purge message # because it is my own
 5:    else if defined message.to ∧ message.to != my-own-name then
 6:       Purge message # not addressed to me
 7:    else
 8:       G.schedule( "process-message", message )
 9: else
10:    Pass
11: G.schedule( "listen", :priority⇒-5, :activation⇒1 )
```

result of this mechanism is that actions with a low priority and a high activation factor will rise relatively fast to the top of the agenda during successive insertions. Conversely, actions with a low activation factor will probably remain on the agenda for a long time, unless they were already given a high priority from the start. Actions that should receive low priorities but high activation factors are typically low-level actions that must be executed on a regular basis "to keep an agent going," without blocking the more important high-level actions. Listening is an example of such an action (and the only example in my model). Conversely, high-level actions, such as logical inference and inquiry, typically receive a high priority but a low activation factor.

3.3 Belief Base

Each agent possesses a private belief base B that is only filled with propositions. A proposition is an object with a number of attributes as described in Table 1.

Propositions can take the form of an atom, a literal, a rule, or the negation of a rule. (The latter is established by naming rules, and then negating the name of the rule.) In the current model the degree-of-belief of a proposition is an element in $[0, 1]$ that indicates to what extent an individual agent believes in that proposition. The degree-of-support of a proposition is an element in $[0, 1]$ that indicates to what extent a proposition receives logical support from other propositions via logical inference. The rules for propagation of support through rules of inference are primitive but provide sufficient material to construct an elementary logic for agents.

A trivial example of a proposition is the atomic proposition P. This proposition has the following slots filled: name, DOB, DOS, supports, supported-by, claimants, questioners, and last-questioned-by. The following slots are empty (and stay empty): antecedent, consequent, strength and RDOS. The rest of the slots are filled optionally.

A non-trivial example of a proposition is the negation of the rule

$$r : P, Q \to S.$$

The slots negation, DOB, DOS, supports, supported-by, claimants, questioners, and last-questioned-by are filled. The negation slot points to the proposition object with name r. The following slots are empty (and stay empty): name, antecedent, consequent, strength and RDOS. The rest of the slots are filled optionally.

Table 1. Proposition

Key	*Description*	*Accessibility*
name	agent's private name for this proposition	optional
negates	reference to proposition that is negated	optional
importance	number that indicates how much importance the agent attaches to the credibility of this proposition	default 1
dob	degree of belief $\in [0, 1]$	default 0.0
DOS	degree of support $\in [0, 1]$	default 0.0
supports	list of references to internal propositions supported by this proposition	default []
supported-by	list of references to internal that support this proposition	default []
claimants	list of agent names that have claimed this proposition	default []
questioners	list of agent names that have questioned this proposition	optional
last-questioned-by	agent that questioned this proposition last	optional
consequent	head of rule	optional
antecedent	body of rule	optional
strength	rule strength	default 1.0
RDOS	degree of support running through this rule	default 0.0

The relation between P and r is that P occurs in the antecedent list of r while r occurs in the supported-by list of P. In the line of Toulmin [29] rules can support or deny other rules. Thus, the consequent of a rule can be another rule, or the negation of another rule (called *undercutter* in Pollock [20] and subsequent work in computational dialectic). (Explained in more detail in overview articles such as [5, 32].)

3.4 The Underlying Argumentation Model

Internally, agents try to enhance their support of selected propositions by means of arguments.

The underlying argumentation model that I use for the larger MAS is a trimmed down version of formalisms as proposed in, e.g., [1, 13, 22, 33]. According to these formalisms, arguments are obtained by chaining rules into trees, and arguments supply different degrees of support to their conclusions. How support is computed depends on the modalities of the various rules and propositions, and how we think these modalities should propagate through an argument. In

Table 2. Message

Key	*Description*	*Accessibility*
id	message id (assigned by table)	optional
from	sender	optional
to	addressee	optional
content type	query | statement	optional
subject	the proposition that the message is about	optional
priority	priority, as perceived by the sender	optional
consequent	if the message is a justification, this field will store the consequent of that justification	optional
antecedent	if the message is a justification, this field indicates its antecedent	optional
strength	if the message is a justification, this field indicates the strength as perceived by the sender	optional
dob	degree of belief as perceived by the sender	optional
DOS	degree of support as perceived by the sender	optional
reference	message to which the present message refers to	optional

reality these modalities are often qualitatively specified ("weakly," "strongly," ..., "certainly") or even plain absent. For the sake of simplicity, my model assumes that modalities are elements of the real interval $[0, 1]$ and that modalities are always present on places where we expect them to be specified.

Definition 1 (Support). *Let σ be an argument.*

1. *If σ is a singleton argument, i.e., if σ is of the form $\sigma = \{p\}$ where p is a proposition, then the* degree of support *of σ is equal to the degree of belief of p:*
$$DOS(\sigma) =_{Def} DOB(p)$$
2. *If σ is a compound argument with conclusion p, top-rule $r : p \leftarrow\!(s)\!\vdash p_1, \ldots p_n$ and sub-arguments $\sigma_1, \ldots, \sigma_n$, then the* degree of support *of σ is given by*
$$DOS(\sigma) =_{Def} \max\{\, DOB(p) \\ \min\{\, DOB(r), s * \min\{\, DOS(\sigma_1), \ldots, DOS(\sigma_n)\}\}\,\} \quad (1)$$

The rationale behind (1) is the so-called *weakest link principle*, which says that every construction (in this case: every argument) is as strong as its weakest link. A convincing justification for the weakest-link principle can be found in work of Pollock [20, 21]. Another principle that I have followed is that rules propagate support with an amount that is proportional to their strength. I immediately admit that (1) is an overly simplistic account of support. Nevertheless, the reason to use (1) is that it provides agents with just enough logical machinery to perform simple defeasible reasoning internally, and to engage in simple dialogues about their own defeasible knowledge externally.

Example 1 (Propagation of support). Consider the following set of propositions.

prop	DOB	*prop*	DOB	*prop*	DOB	*name*	*rule*	DOB
a		b	0.7	c		r_1:	$a \leftarrow\!(0.5)\!\vdash b, c$	1.0
d	0.2	e		f	0.8	r_2:	$b \leftarrow\!(0.5)\!\vdash d, e$	1.0
g	0.1	h	1.0	i	0.6	r_3:	$c \leftarrow\!(0.5)\!\vdash f, g$	1.0
j	1.0					r_4:	$e \leftarrow\!(0.5)\!\vdash h, i$	1.0
k	0.8					r_5:	$g \leftarrow\!(0.5)\!\vdash j, k$	1.0

Thus, we have eleven atomic propositions and five propositions of type rule, or simply: rules. If all reasons are chained together, we obtain a representation of an argument as displayed in Fig. 3. Agents spend resources in trying to discover such arguments through backward chaining and to propagate support forwards (upwards in Fig. 3) in case they receive new information about the credibility of a specific proposition.

How agents schedule inference and communication actions that relate to support is further explained in Sec. 4.2.

4 Actions in More Detail

In the current implementation agents can schedule and execute approximately forty-five different actions, ranging from actions that are concerned with internal inference to actions that are concerned with communication. This section discusses the interaction between the different types of actions and explains how this interaction shapes the discussion. For reasons of space, I do not review all actions.

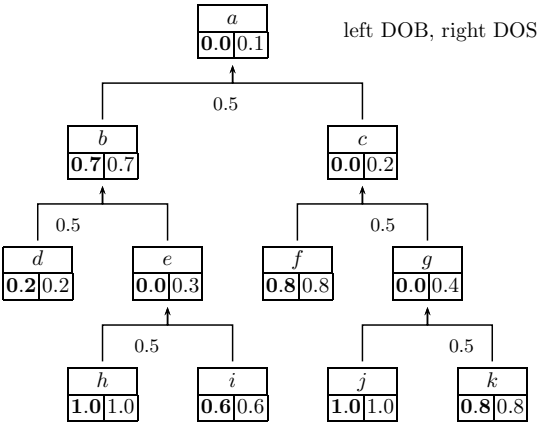

Fig. 3. Propagation of support

Roughly there are three categories of actions. The first category of actions is epistemic and is concerned with inquiry prioritization (which propositions to investigate next), logical inference, and belief updates. Examples of such actions are "propagate-degree-of-support-for p" or "compute-degree-of-support-for p," where p is a proposition. Other actions relate to the external world and are concerned with speaking and listening. A third category of actions is concerned with linking the external to the internal world. Examples of these type of actions are actions to process or incorporate messages and translate them to proposition objects.

An important property of the model is that agents decide autonomously which issues need to be discussed, when to query other agents about issues, when to keep on querying other agents, and when to settle for an answer.

4.1 Inquiry

New queries arise due to a combination of importance and epistemic dissonance [4, 24]:

$$\textit{urgency-to-enquire}(p) = \textit{importance}(p) * \mathrm{DOS}(p) * \mathrm{DOS}(\neg p) \qquad (2)$$

In general, epistemic dissonance is the degree of conflict between two or more competing beliefs, of which at least one belief is deemed important for reasons that may be external to the logical or epistemological formalism (for example for practical reasons) [9, 19]. Here, the epistemic dissonance of a proposition p is simplified into a simple mathematical product.

In the present model, importance is an external factor that, if absent, defaults to 1.0. The principle of epistemic dissonance can be used as a threshold to decide whether it is allowed to query others if there is doubt concerning a proposition that cannot be resolved on the basis of an agent's private beliefs. For example, if the threshold is set to 0.8 then propositions are queried once *urgency-to-enquire*$(p) > 0.8$. This leads to an elementary Action 2.

Action 2. Inquire(p: proposition, i: priority)

1: G.schedule("compute-degree-of-support-for",
   p :priority⇒ $i + 1$, activation⇒1)
2: G.schedule("compute-degree-of-support-for",
   $\neg p$:priority⇒ $i + 1$, activation⇒1)
3: **if** *urgency-to-enquire*$(p) \geq 0.8$ **then**
4: G.schedule("query", p :priority⇒ i, activation⇒1)

The priority settings in Action 2 enforce that the urgency-to-enquire is computed only after the agent did an internal search into its own beliefs on the credibility of that proposition.

4.2 Inference

Inference amounts to all actions that are internal to an agent and are aimed to enhance the degree of support of propositions.

It must explicitly reiterated here that the underlying agent object logic is extremely simplistic and only serves as a vehicle to demonstrate what agents can do (and are supposed to do) if they engage in a discussion. More mature theories of belief revision are to be found in philosophical logic [10] and the theory of Bayesian belief updates [18].

Basically, there are two categories of inference actions, namely *pull* and *propagation* (or: pull and push).

Propagation is best explained in terms of belief updates related to incoming messages. If an incoming message on a proposition p reports on a higher degree of belief in p, then the receiving agent schedules an update to the degree of support of its internal representation of p (remember that agents are credulous). This update amounts to serially propagating the new degree of belief via (internal) rules to other (internal) propositions. The actual propagation is scheduled as well, so that propositional belief updates are interleaved with other actions. This all depends on the priorities that are attached to the belief update actions. Thus, it may well happen that an agents accidently reports misinformation because it has it given a low priority to its internal belief update actions. (This behavior can occasionally be enforced in the implementation by setting the start priority of belief updates to a low values.)

Belief pull corresponds to the informal question "what do I actually know about p?" and is the result of an inquiry action (Action 2). Belief pull comparable to backward chaining, with the restriction that agents must at each cycle decide whether to search further backwards for justification, or to execute other actions first. The present model solves this by attaching a priority to every backward chaining action that is a function of the priority of the original action and the expected maximal return of support.

4.3 Query

A query is a request for information about a particular proposition. Queries can be open or addressed to a particular agent.

Open queries have no explicit addressee and can be taken up by any agent that finds it important enough to process it. When an agent decides to query other agents (compare Eq. 2), it composes a message with the name of the proposition and a token indicating that the message emitted is a query.

An addressed, or directed, query is a request to a specific agent to explain or justify a certain claim. The present model works with open queries only. On the basis of the message format and the deliberation cycle mechanism it is safe to predict that the existing model can be naturally extended to an agent model in which agents know how to deal with addressed messages.

4.4 Response

Two types of messages can appear on the table, viz. queries and statements. Queries have been discussed above.

A statement is simply a public announcement of an agent in which it declares that it believes in a certain proposition to a certain degree of belief. Analogous to queries, claims can be addressed to a particular agent, typically as an answer to a previous query. Alternatively, claims can be addressed to no agent in particular. Such open claims can be seen as theses, or positions, meant to lure other agents into a discussion. The present model works with directed statements only. Further, the present model allows agents to update their beliefs with statements (answers) that are directed to other agents. This possibility to overhear messages

that are aimed at other agents and to respond to such messages is a more or less arbitrary commitment of the architecture.

The three essential actions in forming replies are Action 3, 4 and 5.

Action 3. Process-query(m: message)

```
1: B.incorporate-query( m )
2: reply = fabricate-reply( m )
3: if defined reply then
4:    G.schedule( "speak", reply )
5:    G.schedule( "process-query", m )
6: else
7:    d = Message.new( :subject→no-answers, :referent→ m )
8:    G.schedule( "speak", d )
```

Action 4. Fabricate-reply(m: message)

```
1: reply = next-unpublished-answer-to( m.subject )
2: if defined reply then
3:    return reply.into-message-format
4: else
5:    return nil
```

Action 5. Next-unpublished-answer-to(s: subject)

```
1: prop = B.prop-retrieve( s )
2: if defined prop then
3:    return prop unless P.published(prop)
4: for rule ∈ B.rule-retrieve( s ) do
5:    return rule unless P.published(rule)
6: prop = B.prop-retrieve( s.negation )
7: if defined prop then
8:    return prop unless P.published(prop)
9: for rule ∈ B.rule-retrieve( s.negation ) do
10:   return rule unless P.published(rule)
```

With Action 3, the query is incorporated in B first. This means that the agent creates a corresponding proposition in B (if such a proposition does not exist yet), stores the name of the agent that queried the proposition and the time t that this particular proposition is queried. If there is an unpublished answer, then the receiving agent schedules a speech act in which it emits an answer and schedules a new action to process this query (for there may be more answers). If there are no more answers left, only a speech act is scheduled in which the agent effectively says that it has no answers, either because it has no answers to begin with, or else because it ran out of answers.

With Action 5, the action P.published/1 is a check on the procedural belief base in which the agent verifies whether an agent (including the agent itself) already has published the proposition in question.

5 Implementation

To allow experiments with different set-ups, and to see whether the generated dialogues make any sense, I have implemented the model in the oo-scripting language Ruby. The purpose of the implementation is to experiment with different inputs and with different parameter settings.

The results experiments can be reproduced with the help of an online prototype of which the URL is given at the end of this section.

5.1 Experiments

This section presents a simple example in which a group argues about the credibility of a certain proposition. The example is simple in that it does not involve negation and auto-inquiry has been turned off for the sake of brevity and readability. Examples with slightly more complex input already stir up an emormous amount of actions and messages, so that the structure of the dialogue becomes lost in the output. The reader is invited to try out more complex input at the URL mentioned above.

Suppose we have three agents, Alice, Bob, and Charles, and suppose that Bob is instructed to issue a query on C (Fig. 4).

Fig. 5 shows a trace of the run. First Bob tries to find out how much C is supported by its own beliefs. Then it decides to ask others about C. This query is not related to previous messages, hence the empty reference `--`. The two other agents process this question and burrow into their own beliefs to discover to what extent they support C themselves. Alice responds with a justification. This justification is received by Bob. Since the justification end in B and Bob has no support for B, Bob decides to query further and ask others about B (line three). Charles explains B with $A \rightarrow B$. Finally, Bob says "ok' at line five because it can connect the antecedent of $A\ 0.7 \rightarrow B$ to its own support for A.

For reasons of space. the results displayed here are rather minimal. The reader is therefore invited to experiment online at `http://www.cs.uu.nl/~gv/code/liberal`. The online prototype is supplied with a Java-doc style documentation and the code itself can be downloaded if desired.

```
Agent Alice
B 0.8=> C

Agent Bob
C?
A 0.9

Agent Charles
A 0.7=> B
```

Fig. 4. Input

```
   Bob thinking, enquire
   Bob thinking, compute_dos_for
   Bob thinking, speak
1. Bob [--]: Why C?
   Charles thinking, handle_question
   Alice thinking, handle_question
   Alice thinking, speak
2. Alice [1]: C, since B
   Bob thinking, incorporate
   Alice thinking, interpret
   Bob thinking, compute_rdos_for
   Alice thinking, handle_question
   Bob thinking, propagate_rdos_of_rule
   Bob thinking, question_antecedent
   Bob thinking, attack_antecedent
   Bob thinking, question_antecedent_element
   Bob thinking, speak
3. Bob [2]: Why B?
   Charles thinking, handle_question
   Alice thinking, handle_question
   Charles thinking, speak
4. Charles [3]: B, since A
   Bob thinking, incorporate
   Charles thinking, interpret
   Bob thinking, compute_rdos_for
   Charles thinking, handle_question
   Bob thinking, propagate_rdos_of_rule
   Bob thinking, question_antecedent
   Bob thinking, attack_antecedent
   Bob thinking, question_antecedent_element
   Bob thinking, speak
5. Bob [4]: Ok
```

Fig. 5. Summary of run

6 Results

During the experiments, I noticed that all discussions terminate. This can be understood as follows. Firstly, a finite number of queries may be linked to a finite number of answers. Further, agents keep an account of which queries they have answered. Since, queries are dealt with at most once, termination is ensured for ach individual agent. Since a MAS contains a pre-determined of agents by definition, eventually termination is ensured for the entire MAS.

I also observed that agents will reach a conclusion on accessible facts within a reasonable amount of turns. This can be explained by the fact that explanations (i.e., explanatory rules) cannot be chained indefinitely. A a consequence each justification has a stopping place, so that agents will either accept facts or abandon search on explained statements within a bounded number of dialogue moves.

Properties such as termination and response are proven formally in [17]. Intuitive results reported there indeed correspond with my model albeit my judgement is based on observation rather than on model analysis. Other results do not correspond to my model, for example that credulous agents can be convinced of everything, even of propositions contrary to their beliefs [17, Prop. 6.8, p. 367].

Even though discussions terminate, I noticed that traces of runs are extremely long, even for trivial input. This observation points to two further research problems.

1. The problem to maintain overview on the activity in a MAS.
2. Estimating the number of actions in a MAS based on the size of the input.

Investigation of these problems falls beyond the scope of this paper, but is briefly discussed in Section 8.

7 Related Work

The term of liberal dispute was earlier coined by Prakken in an article on relating protocols for dynamic dispute with logics for defeasible argumentation [23]. In Prakken's work, a liberal dispute is an exchange of arguments (rather than an exchange of propositions as is done in this paper) such that every move is relevant (in Prakken's sense) to the first argument in that dispute. The main effort in Prakken's work is to prove that liberal protocols are sound and fair. It is possible to prove such a result because the formalism assumes that arguments are exchanged in their entirety and that participants in a discussion eventually respond to all utterances that are logically connected to their beliefs. In turn, these assumptions rest on the hypothesis that agents are logically omniscient, communicate everything they know and are able to process everything they receive. The model presented in this paper is less idealistic and thus cannot guarantee such a result.

Although it is arguably one of the simpler types of dialogue, inquiry has received less attention than negotiation or persuasion. An exception is the work by McBurney and Parsons [14] on scientific investigation. Our purpose is very similar to theirs. They describe a *Risk Agora*, as they call it, that allows the storage of multiple arguments for and against some claim. However, they do not treat multi-party issues explicitly. The Agora is an asynchronous channel; no coordination rules are given.

My present work also relates to the Newscast protocol [30]. The Newscast protocol is a kind of 'gossiping' protocol that can be used to disseminate information in distributed systems. A difference is that the newscast protocol can only pass on information. No mechanism exists to specify queries. The Newscast protocol is also implemented and experimented with albeit on a much larger scale, and the results are reported quantitatively.

Recently, researchers in the European SOCS project proposed a model of agency for global computing called the KGP model (knowledge, goals and plans) [3, 26]. This model is particularly interesting because a number of researchers

that worked on this model have a strong background in argumentation. The KGP model proposes a logical architecture that is concerned with agents that (for various reasons) have incomplete information about their environment, and want to update that information by engaging in a conversation with other agents. Like the model that is proposed in this paper, KGP uses uses priorities by defining preference policies over the order of application of transitions. However, the prioritization is more complex because entire logic programs are prioritized rather than atomic actions. Every KGP-agent contains an argumentation component that is a direct derivative of the classical argumentation theories that have preferred and admissible sets as their semantics. It is remarkable that, in other publications, some of these authors argue that finding admissible and preferred arguments can be very hard [7].

The lightweight version of 3APL, called 3APL-M does have a so-called plan ranker [6]. This an internal class, part of the planner sub-system, which classifies the plans in the plan base by calculating its utilities. This component drops the plans that have negative utility from the Plan Base.

8 Future Work

A problem that I noted with ourthe experiments is that it is difficult to monitor all the action. At present all activities are written to a linear log but this solution is unsatisfactory from multiple viewpoints, even for small input. Although there exist tools to monitor agent communication (e.g., JADE's message sniffers [16]), a larger problem is to monitor all pre-processing prior to message emission and all processing of messages once they are received. Currently, I have colored the output to create a global distinction. Each agent possesses its own color. Dark colored log entries relate to internal processing, while light colored log entries relate to agent activity that are more related to communication. Currently there are four such color categories.

9 Conclusion

In this paper I proposed a liberalized version of existing argumentation protocols. Within the resulting setup agents can construct arguments autonomously by participating in an inquiry dialog, thus bringing ideas of computational dialectic to bear in a multi-party inquiry. It is the connection between the two disciplines that counts here. Obviously more work has to be done to consolidate and utilize this connection.

Acknowledgement. I'd like to thank Martin Caminada, Mehdi Dastani, Henry Prakken and two anonymous referees for their helpful comments. This research was supported in part by a European Commission STReP grant ASPIC IST-FP6-002307. This project aims to develop re-usable software components for argumentation-based interactions between autonomous agents.

References

1. P. Baroni, M. Giacomin, and G. Guida. Extending abstract argumentation systems theory. *Artificial Intelligence*, 120(2):251–270, 2000.
2. Robbert-Jan Beun. On the generation of coherent dialogue: A computational approach. *Pragmatics & Cognition*, 9(1):37–68, 2001.
3. Andrea Bracciali, Neophytos Demetriou, Ulle Endriss, Antonis Kakas, Wenjin Lu, Paolo Mancarella, Fariba Sadri, Kostas Stathis, Giacomo Terreni, and Francesca Toni. The KGP model of agency for global computing: Computational model and prototype implementation. In *Proc. of the Global Computing 2004 Workshop*, volume 3267 of *LNCS*, pages 342–369. Springer Verlag, 2004.
4. Urszula Chajewska and Joseph Y. Halpern. Defining explanation in probabilistic systems. In *Proc. of the 13th Conf. on Uncertainty in Artificial Intelligence*, pages 62–71, 1997.
5. Carlos I. Chesñevar, Ana G. Maguitman, and Ronald P. Loui. Logical models of argument. *ACM Computing Surveys*, 32(4):337–383, 2000.
6. Mehdi Dastani, Frank de Boer, Frank Dignum, and John-Jules Meyer. Programming agent deliberation: An approach illustrated using the 3APL language. In *Proc. of the Second Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'03)*, 2003.
7. Yannis Dimopoulos, Bernhard Nebel, and Francesca Toni. Finding admissible and preferred arguments can be very hard. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 53–61. Morgan Kaufmann, 2000.
8. Sylvie Doutre and Jérôme Mengin. On sceptical vs. credulous acceptance for abstract argument systems. In *Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004)*, pages 134–139, 2004.
9. N. Everitt and A. Fisher. *Modern Epistemology: A New Introduction*. McGraw-Hill, 1995.
10. Peter Gärdenfors. *Knowledge in Flux: Modelling the dynamics of epistemic states*. MIT Press, London, 1988.
11. Charles L. Hamblin. Mathematical models of dialogue. *Theoria*, 37:130–155, 1971.
12. Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer. Agent programming in 3apl. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.
13. Fangzhen Lin and Yoav Shoham. Argument systems: A uniform basis for nonmonotonic reasoning. In R.J. Brachman, H.J. Levesque, and R. Reiter, editors, *Proc. of the 1st Int. Conf. on Knowledge Representation and Reasoning*, pages 245–255. Morgan Kaufmann Publishers, 1989.
14. Peter McBurney and Simon Parsons. Representing epistemic uncertainty by means of dialectical argumentation. *Annals of Mathematics and Artificial Intelligence*, 32(1):125–169, 2001.
15. Peter McBurney, Simon Parsons, and Michael Wooldridge. Desiderata for agent argumentation protocols. In *Proc. of the First Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 402–409. ACM Press, 2002.
16. Pavlos Moraitis and Nikolaos I. Spanoudakis. Combining gaia and jade for multiagent systems development. In *Proc. of the 17th European Meeting on Cybernetics and Systems Research (EMCSR 2004)*, April 2004.
17. Simon Parsons, Michael Wooldridge, and Leila Amgoud. Properties and complexity of some formal inter-agent dialogues. *The Journal of Logic and Computation*, 13(3):347–376, 2003.

18. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, Inc., Palo Alto CA, 2 edition, 1994.
19. J.L. Pollock. *Knowledge and Justification*. Princeton University Press, 1974.
20. John L. Pollock. *Cognitive Carpentry. A Blueprint for How to Build a Person*. MIT Press, Cambridge, MA, 1995.
21. John L. Pollock. Implementing defeasible reasoning. Presented at the Computational Dialectics Workshop, at FAPR'96, June 3-7, 1996, Bonn. Cf. http://nathan.gmd.de/ projects/zeno/fapr/programme.html., 1996.
22. H. Prakken and Gerard A.W. Vreeswijk. Logics for defeasible argumentation. In D.M. Gabbay et al., editors, *Handbook of Philosophical Logic*, pages 219–318. Kluwer Academic Publishers, Dordrecht, 2002.
23. Henry Prakken. Relating protocols for dynamic dispute with logics for defeasible argumentation. In Shahid Rahman and Helge Rückert, editors, *New Perspectives in Dialogical Logics*, volume 127, pages 187–219. Synthese, 2001.
24. Anand S. Rao. Integrated agent architecture: Execution and recognition of mental-states. In *Intelligent Agent Systems: Theoretical and Practical Issues*, volume 1087 of *Lecture notes in computer science*, pages 159–173. Springer-Verlag, Berlin, 1996.
25. John R. Searle. Conversation. In J.R. et al. Searle, editor, *(On) Searle on Conversation*, pages 7–30. John Benjamins, 1992.
26. Kostas Stathis, Antonis Kakas, Wenjin Lu, Neophytos Demetriou, Ulle Endriss, and Andrea Bracciali. Prosocs: A platform for programming software agents in computational logic. In J. Müller and P. Petta, editors, *Proc. of the 4th Int. Symposium "From Agent Theory to Agent Implementation" (AT2AI-2004)*, April 2004.
27. Andrew S. Tanenbaum. *Operating Systems: Design and Implementation (Second Edition)*. Prentice Hall, 1997.
28. Jasper A. Taylor, Jean Carletta, and Chris Mellish. Requirements for belief models in co-operative dialogue. *User Modelling and User-Adapted Interaction*, 6:23–68, 1996.
29. Stephen Toulmin. *The Uses of Argument*. Cambridge University Press, 1985.
30. Spyros Voulgaris, Márk Jelasity, and Maarten van Steen. A robust and scalable peer-to-peer gossiping protocol. In *Proc. 2nd Int. Workshop on Agents and Peer-to-Peer Computing (AP2PC 2003)*, 2003.
31. Gerard Vreeswijk and Joris Hulstijn. A free-format dialogue protocol for multi-party inquiry. In Jonathan Ginzburg and Enric Vallduví, editors, *Proc. of the Eighth Int. Workshop on the Semantics and Pragmatics of Dialogue (Catalog '04)*, pages 273–279, 2004.
32. Gerard Vreeswijk and Henry Prakken. Credulous and sceptical argument games for preferred semantics. In Ojeda-Aciego *et al.*, editor, *Proc. of the 7th European Workshop on Logics in Artificial Inteligence (JELIA 2000)*, volume 1919 of *LNCS*, pages 239–253. Springer-Verlag, 2000.
33. Gerard A.W. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90:225–279, 1997.
34. Gerard A.W. Vreeswijk. Liberalizing protocols for argumentation in multi-agent systems. In *Proc. of the 4th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1259–1260, New York, NY, USA, 2005. ACM Press.