Towards a Test Bed for Multi-Party Dialogues

Frank Dignum, Gerard A.W. Vreeswijk
Department of Computer Science
Utrecht University
[dignum,gv]@cs.uu.nl

Abstract

In many situations conversations involve more than two parties. However, very little research has been done yet on modelling multi-party dialogues. In this paper we first explore a number of issues that arise when considering dialogues between more than two parties. Then we take some steps towards creating a test bed in which these issues can be explored and theory on multi-party dialogues can be developed.

1 Introduction

In the past few years quite some research has been done in the area of agent communication (see e.g. [4, 3]). In most of the reported work the dialogues only involve two parties at the time. Even though specifications like the FIPA ACL [6] permit a message to be sent to more than one addressee, the protocols they give are mainly based on two party dialogues. If more parties are involved this is mostly through a broadcast message from one party to all other parties, after which each of these parties reply to the original sender (cf. for instance the Contract Net protocol). Actually, this does not constitute a multi-party dialogue but a number of parallel two-party dialogues.
Much of the work done on agent conversations, such as [14, 15] draws from the work on dialogue theory in linguistics (e.g. [19] is quite influential in this area). The theory gives a typology and theory for the moves that can be made in the different types of dialogues. However, this theory is built for two-party dialogues and there is little or no (usable) research done on multi-party dialogues. Neither in the field of pure linguistics nor in the field of distributed AI did we find much leads to use for constructing theories of multi-party dialogues for agents. The only work we have been able to find treating multi-party dialogues is from Traum [5] which mainly deals with focus of attention and initiative in multi-party conversations.

In practice the need for multi-party dialogues becomes more and more apparent. For instance, several agents have to cooperate in order to find a solution for a problem. Each of the agents might have a part of the solution, but only their interaction might reveal how to combine all the pieces of the puzzle. This might happen when a user agent tries to compose a holiday to a certain town and needs a flight and a hotel and maybe some local entertainment. If each of these components is delivered by a different source maintained by a different agent, it might be beneficial if hotel and flight manager can negotiate dates on which both flights and hotel are available. The user agent could listen in and intervene whenever the dates drift of too much from the preferred dates or the prices get too high. This would be more efficient than the user agent trying to first reserve a flight and afterwards a hotel (or the other way around) and having to go back and forth whenever the hotel is not available on the days between the flights.

In this paper we will explore the field of multi-party dialogues. First, in section 2, we discuss a number of the issues that arise when changing from a two-party situation to a multi-party situation. So, it does not contain a list of all dialogue issues, because many issues exist for both two-party as well as for multi-party dialogues (e.g. whether parties are cooperative or not or are sincere or not). After this exploration of issues, we will present a first implementation of a kind of test-bed in which a number of these issues can be examined in section 3 and 4. We only show the most simple case of an inquiry dialogue. The value of the example shown in section 5 is mainly as a first step of a systematic way of exploring the influence that design choices have on the resulting dialogues.
Whereas the proof of interesting properties of two-party dialogues is already quite difficult (see [14]) it will be even harder in multi-party dialogues. The implementation presented in this paper can be seen as the start of a test bed in which such properties can be empirically explored and theory can be developed towards making the right choices for dialogues depending on the environment of the system.

2 Issues in multi-party dialogues

Probably one of the few things that does not change when moving from two to multiple parties is the typology of the dialogues. The well-known typology of Walton and Krabbe [19] is based on the goal of the dialogue (synchronization of believes and/or actions) and the starting situation (conflicting believes and/or intentions). Both elements play an equal role in multi-party dialogues.

2.1 Open vs. closed systems

The first difference that comes up right away, is whether the system is open or closed. I.e. are all parties present during the whole dialogue or can they join later and/or leave before the end of the dialogue? When we have only two parties there is no choice, both parties are needed to keep the dialogue going. However, when more parties are involved this is no longer necessary. In news groups it probably is more common for parties to join and leave during a dialogue than for parties to stick around during the complete dialogue. Also in situations where the parties consult experts during the dialogue to explain issues or arbitrate on conflicts, the expert is only part of a small slice of the dialogue.
Of course open settings make it very difficult to check whether, for instance, the goal of a dialogue has been reached. Is there general agreement on a course of action, is there mutual believe, is everyone convinced? For each of these questions one should specify which parties (still) count. Another question becomes how we know that a party has left the dialogue or joined it? Are there special speech acts to denote these acts, do participants register?

2.2 Roles

A following issue is the role of each of the parties in the dialogue. This can be viewed from different perspectives. In the first perspective we look at roles from a linguistic point of view. In a two-party dialogue there is always a speaker and an addressee (or hearer). However, in a multi-party dialogue we can at least distinguish: speaker, addressee, auditor, overhearer and eavesdropper (see e.g. [12]). Although the speaker directs a message to one (or more) other parties there can also be parties that hear the message without being addressed explicitly. The auditor is a party that is supposed to hear the message (this is intended by the speaker). An example is when I cc a mail to my boss to "proof" that I indeed made an inquiry as I had promised to my boss before. The overhearer is allowed to hear the message but the speaker does not intend him to hear it. This happens when I send a message as a response to a question in a news group. All people subscribed to that news group can see the response but are not necessarily the intended audience. Finally, the eavesdropper is a party that happens to hear the message without the speaker wanting him to hear it at all. E.g. I do a "reply-all" while I intended a "reply" to a mail. When determining the effect of a message on the other parties one has to take the role of that party into account. A request to perform an action might lead to an intention in the addressee while leading to a dropping of the same intention in the overhearer.

One can also look at the roles from a dialectic perspective. In a typical two-party dialogue there is a proponent and an opponent. In a multi-party dialogue we can also have roles such as: neutral party, interested party, interviewer, advocate, respondent, examinator, challenged party, mediator, or arbitrator (to coin a few disciplinary-neutral terms). The role an agent plays influences the type of responses it can give during the dialogue. E.g. a mediator might give an alternative proposal or might ask for additional information from other parties after the first arguments have been exchanged.

A third type of roles that can be distinguished are the social roles within the dialogue. A good example is that of a chairperson. These types of roles can influence the turn taking within the dialogue, but can also determine when parties can join the dialogue and leave it again. Finally, the chairperson might have the power to terminate a dialogue one-sided or through some predetermined protocol.

A last perspective on roles that we mention here are interests. Some parties in a dialogue will have the goal to terminate the dialogue successfully, while other parties might want just to disrupt the dialogue or try to extend it eternally. This might happen in multi-party negotiations. Some parties might want to conclude the negotiations, trying to get the optimum result for themselves, while other parties benefit most when no agreement is reached at all.

For each of the perspectives on roles one can choose whether roles are fixed once or can change during the dialogue. For linguistic roles this seems the most likely choice, but also other roles might change (either explicitly or implicitly) during a dialogue.

2.3 Medium and addressing

This issue ties in with that of linguistic roles. The main question is how messages are addressed. We can make a distinction between one-to-one distribution, one-to-many distribution and one-to-all distribution. In a two-party dialogue all messages are directed at the "other" party. However, in a multi-party dialogue one can choose whether to address a message to a specific other party or to several (specified) other parties or just broadcast the message to all other parties.

Independently one can decide whether all communication is "overheard" by all parties or not. I.e. can all parties hear all communication or only the messages that are directed to them? One might argue that messages that are only heard by a subgroup are a separate dialogue between that subgroup. In that case all messages should at least be heard by all participants of the dialogue.

2.4 Coordination

The first question is whether the parties can all react asynchronously or whether each time only one party can have its turn. Although the asynchronous coordination may seem chaotic, it is the standard for most news groups and mailing lists.
The issue of turn-taking is relatively trivial in two-party dialogues. However, how is the order determined in which the parties take turns in a multi-party dialogue? Is it a round-robbin protocol (the generalization of the strict turn-taking for two parties) or do we use other mechanisms? As said above, one might also have a chairman that explicitly determines the next party that can or should take its turn.

Independently one should decide on which messages each party can react. Can parties react on all messages delivered before, the last messages of all parties delivered before, only to the originator, etc.? Can they react to a message of which they were not the addressee?

Of course, one should also consider *how* all parties can react. A common rule in many two-party argumentation systems is that parties are not allowed to repeat an argument in order to avoid infinite regression of arguments (see e.g. [16]). In multi-party arguments the question arises whether other parties are allowed to repeat an argument. One could argue that this indicates additional support for an argument and it would be useful to strengthen an argument.

2.5 Termination

Although we argued above that the goals of the multi-party dialogues are similar to the two-party dialogues they have to be translated to their multi-party versions. For instance, does a persuasion dialogue ends successful if all parties are convinced of an argument, or if most are convinced of it, or if some designated parties are convinced? One might say that the original addressees should be convinced while the auditors and overhearers do not have to be convinced. But many more choices are possible.

In the same vein the termination of the dialogue becomes less obvious. Who determines whether a dialogue is terminated? In case there is a chairman he might decide. However, it might also be that a majority of the parties should agree that the dialogue is terminated.

2.6 Properties

There are some interesting new properties to be checked for multi-party dialogues. E.g. can we guarantee that an inquiry dialogue protocol will deliver the answer if the union of all the knowledge of the parties in the dialogue would be enough to derive this answer?
Other issues would involve whether a protocol only reveals information to participants that they need to know in order to respond or whether information is released that parties would rather not divulge if not needed.
Of course the question of guaranteed termination of a protocol is also very relevant for a multi-party dialogue.

2.7 Internal operation

Besides the issues described above on the external properties of a multi-party dialogue one might also want to consider whether the parties should have some extra internal properties to take part in a multi-party dialogue. One obvious candidate is how agents determine when and with what content to respond to which other parties.

The issues discussed in this section probably do not cover the whole field, but hopefully give a good insight in the landscape of multi-party dialogues and its challenges. Far from trying to answer all of the above questions in the rest of this paper we will sketch a framework in which these questions can be studied and possible answers formulated in an empirical way.

3 A starting system: blackboard dialectic

As said before the basis of agent conversations can be found in linguistics and one of the most relevant areas is that of computational dialectics. So, we will take the work from this field as a basis for our own experimental work and discuss the methods of computational dialectic [1990-now] in the next section.
Because multi-party dialogues require different methods of communication than only the one-to-one and one-to-many we decided to use the blackboard system metaphor as a basis for communication. Blackboard systems are probably the most generic form of multi-party communication. More sophisticated communication forms can easily be build on top of these systems. We discuss the methods of blackboard systems [1975-now] in the second subsection below.

Computational dialectic. Influenced by work on nonmonotonic reasoning and argumentation theory, computational dialectic emerged as an area that makes significant progress with the (formal) synthesis and (computational) execution of artificial argument and dispute [10, 18, 1]. As far as we know, however, most if not all work in computational dialectic models disputes in which two parties, named PRO and CON, exchange arguments in such manner that each party is obliged to immediately respond to moves of its (or his, or her) opponent. In his pioneering paper "process and policy," Loui termed this type of dispute as so-called *two-party immediate response dialectic* [11]. An example of a two-party immediate response dispute, adapted from [17], is the following example

Example. Two parties named PRO and CON, have arguments of the most simple sort for and against A, respectively.

Proposition	DOB	Proposition	DOB
B	1.00	B –(0.97)→ A	1.00
C	1.00	C –(0.91)→ $\neg A$	1.00

Thus, both believe all four propositions B, C, B –(0.97)→ A, and C –(0.91)→ $\neg A$ to be absolutely true. This certainty is expressed as a degree of belief (DOB), of 1.00. The implicational strength with which the two rules imply their conclusion is 0.97 and 0.91, respectively. Accordingly, the dispute evolves as in Table 1.

1.	0,0 pro:	I claim that A holds.
2.	1,0 con:	Why?
3.	0,0 pro:	Because it follows from B with strength 0.97, and I claim that B holds.
4.	1,1 con:	Why?
5.	0,1 pro:	Simply because B is the case with DOB = 1.00
6.	1,1 con:	Frankly, I am willing to contest B: I claim that $\neg B$ holds.
7.	2,1 pro:	Why?
8.	1,1 con:	Um... upon closer inspection I see I have no grounds for $\neg B$.
9.	0,1 pro:	That leaves me with B, obviously. And via B my A is supported 0.97
10.	1,0 con:	Sure, but now this: I claim that $\neg A$ holds.
11.	2,0 pro:	Why?
12.	1,0 con:	Because it follows from C with strength 0.91, and I claim that C holds.
13.	2,1 pro:	Why?
14.	1,1 con:	Simply because C is the case with DOB = 1.00
15.	2,1 pro:	Frankly, I am willing to contest C: I claim that $\neg C$ holds.
16.	3,1 con:	Why?
17.	2,1 pro:	Um... upon closer inspection I see I have no grounds for $\neg C$.
18.	1,1 con:	That leaves me with C, obviously. And via C my $\neg A$ is supported 0.91. Nevertheless, I grant that earlier on (line 9), you were able to support A more strongly. Therefore I drop $\neg A$.
19.	0,0 pro:	Combining my support (line 1-9) and your counter-support (line 10-18) yields 0.97 for A, which means that I am right on A.

Table 1: Output of a simple two-person immediate response dialogue.

The first row of numbers in Table 1 are line numbers. Of the second row of numbers (separated by a comma) the first row indicates the *level* of the dispute, i.e., the number of times that the burden of proof has alternated. For example, if PRO starts a dispute on A and, *within* this dispute, CON starts a dispute on B and, within CON's dispute, PRO starts a dispute on C, then the dispute is at level three. The second half of the row of numbers that are separated by a comma indicates the *depth* of the dispute, i.e., the number of times that the defending party justifies his claim via regression through rules.

The point of this sample-dispute is to note that a two-party immediate response protocol forces both parties to enter and explore every part of the search space in a strict depth-first search fashion, something that is not akin to informal dispute. Still, the two-person dispute system presented in [17] provides a number of clues to multi-party dialogue. Therefore, the example implementation presented in this paper continues to build on the system presented in [17].

Blackboard systems. Another development relevant to multi-agent dispute are so-called blackboard systems [2, 7, 8]. We see blackboard systems as the most basic form of communication medium. Many other systems can be seen as refinements of this system. A blackboard system is an expert system based on the blackboard metaphor. This metaphor says that, to solve a problem, or to answer a question, a number of more or less independent artificial experts, named *knowledge sources* in BB-terminology, should communicate by means of a central medium, *the blackboard* from which all knowledge sources may read and write. Of course, there must be some protocol to arrange who goes to the blackboard if more than one expert “reaches for the chalk”, and there must be agreement on a common language that experts use, but these questions are clearly addressed in research on blackboard systems. In the initial stages of research on knowledge based systems, the motivation to work with blackboard was that such systems are less compiled and more modular than conventional expert systems. The idea was that they should be more transparent and easier to maintain than monolithic expert systems. After some time, however, research on blackboard systems lost focus, because it was claimed that they were not fast enough to meet the DARPA criteria at that time [9, p. 350]. (An argument that has now become obsolete.)

In 2003 it is fair to state that research on multi-agent systems has turned the blackboard metaphor into something that does not live up to contemporary standards of agent-communication and geographically distributed expertise. However, research on blackboard systems can be put into new light by elaborating on the idea of a newsgroup expert discussion. For example, if a computer programmer has a question on feature X of the newest release of programming language Y, he or she can post a question to the appropriate newsgroup and may expect an answer within a time range of one to twenty-four hours (depending on the turnover of a particular group). More specifically we propose to convey the idea of cooperating agents from internet metaphor such as the newsgroup metaphor or, in Google terminology the *discussion groups* metaphor, to a simple but new protocol for the exchange of knowledge. Our motivation for this step is that news and discussion groups are the *de facto* standards to let human agents communicate asynchronously in public. The example implementation presented in this paper adopts the blackboard metaphor (rather than adopting its precise architecture) and gives it a dialectic twist.

4 Implementation

In this and the following section we describe the implementation of a dialectic blackboard architecture implemented in Ruby. We choose for Ruby because it is a pure object-oriented scripting language with an intuitive syntax, suited for prototyping. (Some even advocated Ruby as executable pseudo code.) Based on two-party disputes in computational dialectic on the one hand, and based on the blackboard metaphor in expert systems on the other hand, we start with a dialectic blackboard architecture with the following multi-party properties:

i. A fixed number of equivalent participants engage in an inquiry dialogue, with the goal to discover how to undertake a particular action. So, in terms of section 2.1 we start with a closed system. We start with inquiry dialogues, because they are the easiest with respect to regulating turn taking.

ii. There are no specific roles for the agents. They are equivalent in all respects.
iii. Agents communicate through a central medium, called the forum, the function of which may be compared to the function of an internet newsgroup. Messages are public. They are not addressed to specific agents.
iv. Agents act (listen, reason, and speak) in turn, for a fixed number of rounds. The current experiment, for instance, involves 3 agents and 7 rounds, which implies the performance of 21 atomic acts.
v. There is no criterion for termination. Cf. point (iv).

The following properties are not typical multi-party issues, but also determine the course of a dialogue.

a. Participants are cooperative. They do not lie about their beliefs. All agents acknowledge and process all messages (in so far these messages are applicable), all agents have ample time to reason, and all agents have the opportunity to post all their messages desired.
b. Agents have logical capabilities. In particular, they do not ask what they already know or can infer. Before asking, an agent tries to infer the desired item itself.
c. The facilitation of information is dialectic: claims are justified with other claims or denied with reasons that support a contradiction. Agents accept claims if and only if they can be resolved to information that they believe to be true.
d. Regression to previous messages is always possible. Agents are allowed to question or justify prior claims. Thus, an immediate response is not required.
e. For simplicities' sake the agents have a shared ontology. One consequence of this assumption is that propositions (internal representations of claims) conveyed through messages are not renamed.

4.1 Notation

We use the following notation as convenient shorthand in our implementation.

Definition 4.1. A *discussion group* $G = \langle \mathcal{A}, F \rangle$ is a (possibly infinite) set of agents $\mathcal{A}$ that communicate by means of a forum F. Individual agents are denoted by a_i.

A *forum* $F = \{m_i \mid 1 \leq i \leq n\}$ is an array of messages, where m_n is the last message published. A *message* $m_i = (q, A)$ is a tuple consisting of a question q, possibly followed by one ore more answers to that question. The set of answers, A, is an ordered list.

The internal structure of agents, questions and answers is left unspecified in the theory, but a possible interpretation is elaborated in the rest of this section. Each agent a_i possesses a name, knowledge, a number of questions to be answered, a bookmark to remember the first unread item, bookmarks per question for the first unread answer of that question, and a hash to remember which questions have already been answered (Figure 1 on the next page.)

4.2 Data structures

The agent's knowledge is implemented as a hash, where each atomic action is mapped onto a set of alternative preconditions, or to the value TRUE. The latter indicates that the agent knows how to perform this action.

```
@name           = 'Pete'
@knowledge      = { 'a' => [['f', 'b'], ['c', 'd']],
                    'd' => [['e', 'b'], ['c']],
                    'p' => TRUE
                  }
@questions      = { 'a' => nil, # new question
                    'd' => 3, # first unread answer
                  }
@fid            = 45 # first unread article (fid means "forum identifier")
@bookmark       = { 4 => 2, 56 => 6 }
@answered       = { 4 => TRUE, 8 => TRUE }
```

Figure 1: Datastructure per agent.

The scenario consists of a (theoretically unlimited) number of agents which run concurrently. Furthermore, the scenario accommodates a forum that is shared by all agents. The forum is a passive medium, but is otherwise responsible for the management and administration of messages and personalities (agent-id's). In our current model, these bookkeeping activities amount to no more than maintaining a hash that maps questions onto message-id's (a so-called *reverted index*).

```
N => { 'owner'    => 'Claude',
       'question' => 'bake a cake',
       'answers'  => [ { 'owner'  => 'Francois',
                         'answer' => 'knead the flower', 'bake the paste' } ]
```

Figure 2: Datastructure per question.

Each agent may be in a consumptive mode or in a productive mode. In the consumptive mode an agent takes actions that are supposed to deal with the accumulation of new knowledge. In this case, we have limited this part of the algorithm to two actions, viz. reading the forum and publishing questions to it. In the productive mode an agent does things that are supposed to deal with the dissemination of (private) knowledge. In our case, we have limited this part to answering questions of other agents. In the following paragraphs, we describe how an agent reads a forum, publishes questions, and responds to questions of other agents.

4.3 Consumptive mode

An agent reads news by cycling through its own (typically short) list of unanswered questions. (Cf. Figure 1.) For each such question, it locates the message-id of this question by means of a reversed index. (Recall our assumption about the administrative responsibilities of the forum-object.) If the question is absent, the agent publishes its question and proceeds with its next (and possibly non-assimilating) activity.[1] Else, if the question is already present in the forum, the agent starts scanning the answers to that question, beginning with the first answer not read (by that agent), and ending with the last answer. The point where the agent left the previous time reading answers to that particular question is called the agent's bookmark for that question. Previous to the first scan of a question, the bookmark is typically set to zero. After scanning all the answers to a question, the bookmark is put behind the last question.

Agent method 1 reading answers

```
Require: index
while @bookmark[index] < public.nr_of_answers(index) do
  question = public.question(index)
  @bookmark[index] += 1
  answer = public.answer(index, @bookmark[index])
  incorporate_justification(question, answer)
  if elementen_uit_verklaring_volstaan?(answer) then
    delete_question(question)
    break
  end if
end while
```

When an agent reads an answer it first incorporates the question/answer-combination into its knowledge base. It then verifies whether the answer really suffices as an answer (a process to be explained shortly). If the answer suffices, the agent deletes the question from its private question list and marks it as answered in the forum. It also stops reading further answers to this question. Else, if the answer does not suffice, the agent proceeds reading answers to that particular question. Notice that an answer is incorporated in the knowledge base, irrespective of whether it suffices as an answer or not. This is because an unsatisfactory answer may become useful in a later stage of the process if further knowledge becomes available.

To verify whether a particular answer is satisfactory, an agent verifies whether all elements of that answer can be reduced to its own knowledge. If one such element cannot be resolved, the verification of the other elements is suspended. This is because answers to dependent issues become irrelevant when the original answer cannot be answered either. To verify whether an element can be reduced to private knowledge, the agent verifies whether it has marked the element as true in its own knowledge base, or whether its knowledge base contains a rule supporting that element. In the latter case, the algorithm recursively applies the verification process until the agent encounters private facts or, in the negative case,

[1] A published question may roughly be compared to what is called a *knowledge source activation record* (KSAR) in blackboard systems. Cf. [2].

cannot further justify one particular element. When a particular element cannot be further reduced (i.e., justified) it is published as a question to the forum.
The last type of action is crucial to the process as it closes the consumption/production cycle of questions and answers.

4.4 Productive mode

When in productive mode an agent reads the news chronologically, starting at the first message that was not read by that agent. Thus every agent now possesses two types of bookmarks: one global bookmark for keeping track of questions, and one for each question to keep track of the answers to that question. Since it does not seem to make much sense to try to answer one's own questions, an agent skips its own messages (i.e., questions) if it is in productive mode. A question is also skipped if an agent has given all its answers to that particular question. Else, it publishes an answer to that question. This answer may not already have been published by other agents, including itself. Answers to questions cannot (and need not) be bookmarked, because other agents may have contributed possibly identical answers in the meantime.
After this action, the agent proceeds with its next (and possibly non-productive) activity.

It turns out that the consumer-part of the algorithm is more complex than the producer part, which might seem to reinforce an old adage, namely, that listening is more difficult than speaking.

5 A sample dialogue

As an example, suppose we have three equivalent parties, Arie, Bill and Claude. Arie stands next to his bike. He has pulled out the front wheel out of the fork and holds it up. Bill and Claude are two friends of Arie and stand next to him. (Cf. Figure 3 on page 15.) Bill's `neg-knowledge` indicates that pulling out a wheel and standing there are perfect conditions to ensure that you can no longer go to the hardware store.

With this information, the following dialogue ensues:

1.	Arie:	asks: "How do I repair a flat tire?"
2.	Bill:	responds: "Arie, if you want to repair a flat tire then you must buy a new tire and apply it"
3.	Claude:	responds: "Arie, if you want to repair a flat tire then you must find the leak and seal it"
4.	Arie:	tries to recall for himself how to buy a new tire ..
5.	Arie:	.. he realizes he does not know how to buy a new tire
6.	Arie:	therefore suspends his attempt to apply it
7.	Arie:	tries to recall for himself how to find the leak ..
8.	Arie:	realizes that he knows how to find the leak
9.	Arie:	knows that find the leak can be done by trying to inspect the tire and then mark the leak
10.	Arie:	tries to find the leak by trying to inspect the tire and then mark the leak

11.	Arie:	tries to recall for himself how to inspect the tire ..
12.	Arie:	.. he realizes he does not know how to inspect the tire
13.	Arie:	therefore suspends his attempt to mark the leak
14.	Arie:	therefore suspends his attempt to seal it
15.	Arie:	asks: ”How do I buy a new tire?”
16.	Arie:	asks: ”How do I inspect the tire?”
17.	Bill:	responds: ”Arie, if you want to buy a new tire then you must go to the hardware store”
18.	Claude:	responds: ”Arie, if you want to inspect the tire then you must put the tire in a bucket and watch for bubbles”
19.	Arie:	tries to recall for himself how to go to the hardware store ..
20.	Arie:	.. he realizes he does not know how to go to the hardware store
21.	Arie:	asks: ”How do I go to the hardware store?”
22.	Bill:	responds: ”Arie, if you want to go to the hardware store then do not pull out wheel from fork and stand there like a looney”
23.	Arie:	knows how to pull out wheel from fork and does so.
24.	Arie:	tries to recall for himself how to stand there like a looney ..
25.	Arie:	.. he realizes he does not know how to stand there like a looney
26.	Arie:	asks: ”How do I stand there like a looney?”
27.	Arie:	knows how to put the tire in a bucket and does so.
28.	Arie:	knows how to watch for bubbles and does so.
30.	Arie:	now inspect the tire by put the tire in a bucket, watch for bubbles first and then inspect the tire
31.	Arie:	does mark the leak
32.	Arie:	does seal it

Table 2: A sample dialogue

The formulation of line 23-26 is somewhat unfortunate, because the knowledge items (as displayed in Figure 3) are actually action descriptions, rather than verifications of the state of an agent. Line 23 actually indicates that the wheel *has* been put out, and lines 25-26 actually indicate that Arie cannot verify (or does not agree) that he looks foolish. Apart from this and similar linguistic glitches (that were to be expected due to interpolating[2] agent-specific strings into different template utterances), we maintain that the dialogue otherwise exhibits a surprisingly natural flow of messages and message justifications, so much so, we think, that the algorithm hints at a further and more mature elaboration.

6 Conclusion

In this paper we have scanned the landscape of multi-party dialogues. Some issues have been discussed that surface when changing from two-party dialogues to multi-party dialogues. Many issues come up that have crucial influence on the ensuing dialogues. In order to get some grip on this area, we started of to implement a multi-party dialogue

[2]Like Perl’s string interpolation: if `$year` is 1984, then `"It’s $year"` evaluates to `"It’s 1984"`.

architecture in which we can systematically research these issues.

We emphasize again that the implemented example protocol is extremely simple and is not designed to compete with established and more elaborated protocols for the exchange of knowledge.

What we do hope, however, is that our example has illustrated the possibility to set up a test-bed on the basis of this implementation in which a large number of parameters can be set and in which different ideas on multi-party deferred-response protocols can be tested and improved.

What we have shown already is how two-person immediate response protocols can be generalized and extended to a multi-agent setting by means of a simple blackboard or forum metaphor.

As next steps we have to look at the way turn taking is organized in the system. At the moment this is coded in the system itself and can't be regulated either by the user or the agents. The next point will be to extend the dialogues to include persuasion dialogues. This is a big step because it involves the inclusion in the agents to process inconsistent information and finding counter-arguments for arguments.
Also interesting is the issue of linguistic roles. Instead of using a plain blackboard structure we can make sections that are visible for different sets of participants. In this way we can simulate different linguistic roles.
Finally, we hope to start looking into properties such as termination of the dialogues depending on these different parameters.

Acknowledgements. We'd like to thank Joris Hulstein for sharing his knowledge with us on collective addressees and audience roles in multi-party dialogue theory.

References

[1] P. Baroni, M. Giacomin, and G. Guida. Extending abstract argumentation systems theory. *Artificial Intelligence*, 120(2):251–270, 2000.

[2] D.D. Corkill. Blackboard systems. *AI Expert*, 6(9):40–47, 1991.

[3] F. Dignum and B. Chaib-Draa. *Computational Intelligence: Special issue on Agent Communication*, 18(2), 2002.

[4] F. Dignum and M. Greaves. *Issues in Agent Communication*, volume LNCS(1916). Springer-Verlag, 2002.

[5] D.Traum et al. Embodied agents for multi-party dialogue in immersive virtual worlds. In *Proc. of the First Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pages 766–773. ACM Press, July 2002.

[6] FIPA. Fipa agent communication specification. http://www.fipa.org/, 2003.

[7] B. Hayes-Roth et al. Blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

[8] B. Hayes-Roth et al. Builing systems in the BB* environment. In R. Englemore et al., editors, *Blackboard systems*, chapter 29. Addison-Wesley, 1988.

[9] P. Jackson. *Expert Systems.* Addison-Wesley, 1999.

[10] R.P. Loui. Defeat among arguments: A system of defeasible inference. *Computational Intelligence*, 3(2):100–106, April 1987.

[11] R.P. Loui. Process and policy: Resource-bounded nondemonstrative reasoning. *Computational Intelligence*, 14(1):1–38, 1998.

[12] R. Malouf. Towards an analysis of multi-party discourse. Talk, 1995.

[13] P. McBurney et al. 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.

[14] P.J. McBurney and S. Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, (to appear).

[15] C. Reed, T. Norman, and N. Jennings. Negotiating the semantics of agent communication languages. *Computational Intelligence*, 18(2):229–252, 2002.

[16] P. Torroni. A study on the termination of negotiation dialogues. In *Proc. of the First Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 1223–1230. ACM Press, 2002.

[17] G.A.W. Vreeswijk. Eight dialectic benchmark discussed by two artificial localist disputors. *Synthese*, 127:221–253, 2001.

[18] G.A.W. Vreeswijk and H. Prakken. Credulous and sceptical argument games for preferred semantics. In Manuel Ojeda-Aciego et al., editors, *Proc. of the European Workshop on Logics in AI*, pages 239–253, Berlin, 2000. Springer-Verlag.

[19] D. N. Walton et al. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning.* State University of New York Press, Albany, NY, 1995.

```
Agent.new(
   'name' => 'Arie',
   'questions' => { 'repair a flat tire' => TRUE },
   'knowledge' => {
      'pull out wheel from fork'    => TRUE,
      'stand there like a looney'   => TRUE,
      'seal it'                     => TRUE,
      'find the leak' => [['inspect the tire', 'mark the leak']],
      'put the tire in a bucket'    => TRUE,
      'watch for bubbles'           => TRUE })


Agent.new(
   'name'      => 'Bill',
   'questions' => {},
   'knowledge' => {
      'repair a flat tire' => [['buy a new tire', 'apply it']],
      'buy a new tire'     => [['go to the hardware store']] },
   'neg-knowledge' => {
      'go to the hardware store' =>
         [['pull out wheel from fork', 'stand there like a looney']] })


Agent.new(
   'name'      => 'Claude',
   'questions' => {},
   'knowledge' => {
      'know that the hardware store closed' => TRUE,
      'repair a flat tire' => [['find the leak', 'seal it']],
      'inspect the tire' =>
         [['put the tire in a bucket', 'watch for bubbles']] },
   'neg-knowledge' => {})
```

Figure 3: Agent creation.