Specifying the Interaction Between Information Sources

T.J.M. Bench-Capon

Department of Computer Science, The University of Liverpool,
Chadwick Building, P.O. Box 147, Liverpool L69 3BX, England.
tbc@csc.liv.ac.uk

Abstract. Recently there have been proposals for sophisticated informations systems which require a number of information sources to communicate with each other autonomously. In this paper we discuss how such interactions might be specified, using performatives specified in terms of preconditions, postconditions and completion conditions, identified with respect to conversation classes. Full specification requires additional conditions deriving from the policies and strategies of particular agents. The result is an approach to the specification of interaction between information sources which is uniform, and so permits generic handling of the different sources, but which provides sufficient flexibility to allow for different styles of conversation, and customised agent behaviour.

1 Introduction

Today, through the Internet and World Wide Web, information is more accessible than ever before. Users can get information from distant databases and other information sources. One influential suggestion for an architecture to support the automated collection of this information is that proposed in Wiederhold (1992), which introduced the notion of a *mediator* as a program which when presented with an information need would locate, extract and combine information from a variety of sources to provide a value added service. There is now a considerable body of work following these aims, perhaps most notably the group of projects which go under the umbrella title of the Knowledge Sharing Effort (Neches et al 1991). Other approaches share many of these ideas, perhaps most generally systems based on co-operating communicating agents (e.g. Wooldridge and Jennings 1995).

In all these systems is that we have a collection of software programs able to communicate with one another under their own control. Since the various systems will have been built in different styles and for different purposes, they must be extended in some way to give them these common communication facilities. The architectural solution in the Wiederhold style of system is to provide each system with a *wrapper*, which supplies common communication protocols, performs language translation and the like. (In an agent system this would correspond to the communication layer, but I shall use "wrapper" henceforth in this

paper, and "agent" to refer to any wrapped resource, with no commitment to what lies behind the wrapper.) When the systems are wrapped in this way they present a uniform appearance to each other: the precise nature of the system is hidden behind the wrapper. In this paper I want to look at how we might specify the communication capabilities of these wrappers.

The main tool for talking about these communication facilities is the notion of a *speech act* or *performative*. The key elements in the theory for our purposes are:

- Different speech acts can use the same propositional content: thus the same utterance can be used both to make an assertion and to ask a question. In speech we can indicate the difference with tone of voice, and in writing by punctuation. In this way we can separate the speech acts from their content, and see the speech acts as the focus with their content as a set of parameters.
- Speech acts are intended to have effects, and are characterised by these intended effects. Thus an assertion is intended to cause the beliefs of its addressee to be modified, while a question is intended to cause the addressee to supply information. Thus when uttering a speech act, the utterer has the intention to bring about some effect, and our linguistic conventions supply the necessary causal link between utterance and desired effect.

Speech acts have become rather dominant in considering how to effect the wrapper layer of the systems we are interested in. The most developed use is in the Knowledge Query and Manipulation Language (KQML), as set out in Finin et al (1994). KQML provides a set of speech acts which are intended to provide an extensible core for communication between the different systems in a knowledge sharing network. We will follow this approach, but focus more closely on how such speech acts can be specified. KQML uses the term *performatives* rather than "speech acts", and we will follow this in the remainder of this paper.

2 Specification of Performatives

The starting point for our investigation is the thesis of Labrou (1996). Labrou provides semantics for KQML performatives by giving, for each performative:

- A natural language description of the performative's intuitive meaning;
- An expression that describes the performative, essentially a formalisation of (1) using certain primitives (discussed below);
- Preconditions which must be satisfied if the sender is to issue the performative and the receiver is to accept it;
- Postconditions that describe the states of sender and receiver after the performative has been issued;
- Completion conditions that indicate the final state (possibly after some additional conversation) after the intention which led to the issue of the performative has been fulfilled;
- Comments to enhance understanding of the description of the performative.

The primitives used in the specification above refer to the cognitive states of agents, and are *believe*, *know*, *want* and *intend*. These primitives obviously rely heavily on the terminology of agents, and might be thought inappropriate to a system which is intended to include more conventional systems such as databases among its communicating systems. If we wish to adopt this style of specification we would need to operationalise these states in the wrapper in a way appropriate to a wrapped system; for example, in the case of a database, *bel(A,P)* means that P is in the database A, whereas *knows(A,P)* means that P is such that its truth or falsity can be confirmed by the database A.

As an example of this style of specification consider Labrou's specification of the speech act *tell*, which has three parameters: the sender, (A); the recipient, (B), and the propositional content, (X).

- A states to B that A believes X to be true.
- bel(A,X).
- Preconditions for A: bel(A,X) and know(A, want(B,know(B,bel(A,X))))
- Preconditions for B: intend(B,know(B, bel(A,X)))
- Postcondition for A: know(A,know(B,bel(A,X)))
- Postcondition for B: know(B, bel(A,X))
- Completion: know(B, bel(A,X))
- The completion holds unless B cannot acknowledge the tell correctly.

There are several points to note about this definition:

- There is some commitment to a particular use of *tell*. The preconditions on A mean that it can only be issued if A thinks B wants to know X, and so cannot be used to broadcast information or to volunteer it in case the recipient might be interested. This already suggests the notion of some conventions regulating the conduct of the conversation, but which are implicit in the semantics as given by Labrou.
- There is no guidance on whether A should issue the performative or not. Here there is a number of issues which need to be considered, such as whether it is in A's interests to tell B that X, whether A has the authority to tell B that X, and the like. Labrou understandably does not consider these issues, but if we are to move to a practical system they become of high importance.
- The primitives require the various agents to have beliefs about the other agents, as well as themselves and the conversation they are engaged in. This is a complicating factor: is it really necessary? In fact, I believe it is not, and will introduce a way of eliminating these iterated modalities in section 5.

Labrou offers us a very promising starting point for specifying performatives. It does, however, leave out a number of factors which need to be specified if we are to equip our agents with the required communication ability. It provides no more than the barest semantics for the performative, does not dictate their use sufficiently in some cases, and perhaps over constrains their use in others. We will consider what more we need to be specify in the next section.

3 Specifying the Use of Performatives

Recent work on dialogues (see Bench-Capon 1997 for an overview) has observed that arguments can be evaluated in several ways; as well as argument being *sound*, we can also consider whether it is *correct*, which is judged by whether it accords to the conventions of the particular argument situation. This is what gives rise to different argument "games", At a third level we can argue *well*; within the rules of a particular type of argument, there may often be choices to be made amongst different moves, and one choice will typically be better than the others. Just as we may play chess in accordance with the rules and lose because we make bad moves, so too we can argue soundly and correctly, and yet fail to persuade because we argue badly.

These points about argument can be generalised to conversation in general. As well as knowing how to speak a language, we must also be aware of the different conventions that apply to different conventions, and how to use the conventions to maximum effect.

The importance of conventions governing the use of performatives is recognised in work such as that on COOL (Barbuceanu and Fox 1995). Their approach is to distinguish a number of *conversation classes*, each characterised by a state transition diagram which reflects the way in which performatives can be issued within the particular conversation class.

This addition, however, suffers from the drawbacks:

- It is not integrated with the specification of the performatives themselves;
- The state transition diagram may reveal a number of choice points at which the conversation rules leave the performative to be issued undetermined. In other words it says only how to play the game, not how to play it well.

From the above discussion, I conclude that in order to be in a position to give a full specification of performatives, capable of being implemented in a determinate manner, we need to specify material at three levels:

- The intrinsic level - corresponding to understanding the performative in general
- The conversation level - corresponding to how the performative *can* be used in conversations
- The heuristic level - corresponding to how the performative *should* be used to make the conversation as good as it can be.

In the next section I shall introduce some motivating examples.

4 Example Information Sharing Conversations

In this section I shall introduce some related conversations, concerned with the obtaining of information from a database. I shall use a state transition diagram notation for the conversation.

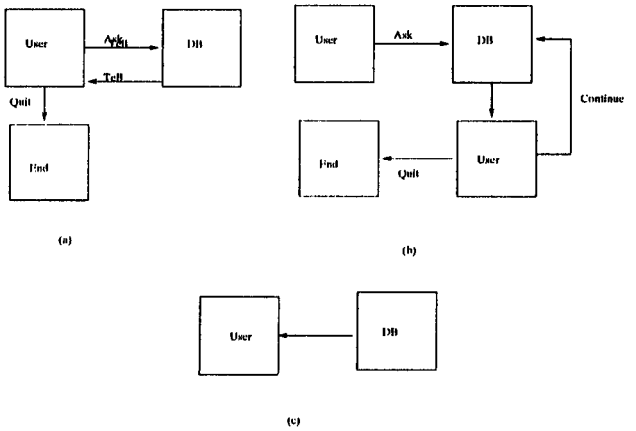

Fig. 1. (a) Conversation 1 - Simple Database Query: (b) Conversation 2 - Query with updates: (c)Conversation 3 - Unsolicited Information

The simplest case is where a user simply queries a database, as shown in Figure 1a.

This is probably what is in mind in Labrou's definition of *tell* given in section 2. However, it is by no means the only possibility for *tell*: consider the following cases, depicted in Figure 1b and Figure 1c. Figure 1b represents a conversation in which once a question has been posed, the user can request up-dates until he chooses to quit, useful, for example, for keeping track of a sporting event which is currently happening, or for share prices. In order for *tell* as required here we need to change Labrou's specification. Now the *tell* is not complete until the user explicitly quits - the query remains live until no longer required. Also it is a further precondition on the issue of the second and subsequent *tells* that such *tells* are only issued when the information changes.

Figure 1c illustrates the case where information is supplied without being solicited. Again this is excluded by Labrou's specification in that the precondition requires A to know that B wants this information. The specification given in section 3 gives only one interpretation of *tell*, and yet the other interpretations seem quite natural.

We can also raise other questions. Suppose for example that A is not permitted to reveal its beliefs to all and sundry, but only if the recipient has appropriate authorisation. Here we would need another precondition on A, namely that B satisfies these authorisation requirements. This is orthogonal to which of the three conversations we are in.

What is emerging from this is that all three elements of good conversation identified at the end of the last section can contribute preconditions, post conditions and completion conditions to the specification of the performative. In the next section I shall consider some of the various elements that could be used in the specification of *tell*.

5 Specification of *tell*

In this section I shall identify some of the various things that might be used to give preconditions, postconditions and completion conditions for *tell*. I shall, in giving these, attempt to eliminate iterated modalities on the cognitive states of agents in favour of conditions expressed in terms which refer to the conversation itself. While it is reasonable to allow, for example, a precondition expressed in terms of the beliefs of the agent issuing the performative (easily operationalised in terms of the response to a particular query), to require the agent to have beliefs about the beliefs of other agents requires considerable sophistication on the part of the agent, and must face problems of uncertain and incomplete information. Moreover, since the application in which we are interested deals with information sources rather than agents, such capabilities go beyond what we would expect of, for example, a database.

In any case, I believe this approach is preferable on other grounds. For example if someone asserts that P, I cannot infer that he believes that P; he may be lying, or introducing it to expose a contradiction, or whatever. I can, however, conclude that *for the purposes of the conversation* he is committed to P. Thus restricting knowledge of other agents to what is manifested in the conversation seems both sensible and right.

To support this we will need to maintain information on several things:

- The performatives issued in the conversation so far
- The commitments of the participants as a result of the conversation so far
- The goals manifested by the agents in the conversation so far. In fact I shall not use goals in this paper, but they are mentioned for completeness.

5.1 Conditions Derived From the Conversation Classes

In this subsection I shall look at the preconditions, postconditions, and completion conditions that are implied by the three conversation classes depicted previously in Figures 1 to 3.

Tell in Conversation 1 In conversation 1 a *tell* can be issued only in response to an ask. Thus letting S be sender, R be recipient and X be content:

```
Precondition: R has issued an ask(X) to S
Postcondition: S is committed to X
Completion: none
```

Note that S need not believe that X; S may have other reasons for giving this reply. Nor need R believe that X or that S believes that X. What R chooses to do with S depends on additional considerations.

Tell in Conversation 2 In conversation 2, S may issue updates to the information supplied. The general question here is X, and X' are successive answers to that question. The conditions for tell(X') are:

```
Preconditions: R has issued an ask(X) to S,
               S has not issued a tell(X'),
               There are no uncompleted tell performatives
Postconditions: S is not committed to any previous answer to X
                S is committed to X'
Completion: R has issued a continue, or R has issued a quit
```

The second precondition, together with the completion condition ensures that the same information is not supplied twice and the third that it is the sender's "turn" to speak. The postconditions replace the commitment to the old answer with the later one.

Tell in Conversation 3 Here the information is unsolicited, so there are no preconditions imposed on *tell* by the conversation.

```
Preconditions: none
Postcondition: S is committed to X
Completion conditions: None
```

Note that here there are no preconditions or completion conditions imposed by the conversation class: this is a one performative conversation, always available to the agent.

Summary of Conditions From Conversation Classes From the above we note that there is only one element common to all three conversation classes, namely that S is committed to X. This does indeed capture the core notion of *tell*, that it is used to commit the speaker to a proposition, for the purposes of the conversation. Making such a commitment is an important feature of any information related conversation, and so we will expect to find some form of *tell* in any such conversation. The remainder of the conditions identify when such a commitment can be made in the context of a particular conversation. By themselves, however, these conditions are insufficient to determine when an agent should issue the performative, and what use will be made of the commitment it brings about. We must therefore now consider supplementary conditions that will be individual to an agent, and derive from policies and heuristics.

5.2 Conditions Derived From Policies and Heuristics

In this section I shall consider a number of possible additional conditions that can supplement those from the conversation class and which will personalise the behaviour of the agents.

Preconditions Consider first a precondition, found in Labrou's specification,

```
believes(S,X).
```

If we have this as a precondition, an agent will only be able to commit to a proposition that it believes to be true. Adding this precondition thus gives us an *honest* agent. Sometimes this may well be desirable, depending on the role of the agent within the overall system. It is not, however, always desirable, particularly in some more complicated conversations in which propositions must be advanced "for the sake of the argument".

Next consider a situation where the agent has sensitive information which can only be issued to a restricted group. Here we will need a precondition expressing that the receiver is allowed to receive the information, for example:

```
permitted_to_receive(R,X).
```

We will need to supply a definition of what it means for a resourse to be permitted to receive a piece of information. For example the information source may maintain a list of registered users, and the precondition would be satisfied if the recipient was on that list. Or the conversation might have been preceded by another conversation in which the recipient supplies a valid password. The details are unimportant; what is important is that we can effect the restriction on information through the precondition.

As an example of a heuristic precondition consider conversation 2. If the information is constantly being updated, we may wish to issue updates at periodic intervals, rather than every time there is a change. This could be effected by a precondition stating that a certain period of time must have elapsed since the last *tell*.

Postconditions Postconditions help us to specify what the effect of a performative is, and this will very often depend on the stance of the recipient. For example, the recipient could also choose to commit to X, in which case we could add the postcondition

```
R is committed to X
```

This gives us a *weakly credulous* agent, which is willing to commit to the proposition for the purposes of the conversation, but which does not modify its actual beliefs. We could, however, give a *strongly credulous* agent, but adding the postcondition

```
believes(R,X)
```

This would cause the receiver to be updated. This would be useful, for example in a stock trading system, which is waiting for the price to reach a certain level before selling. In order to determine this, the information received must be believed so the appropriate tests can be applied, and this information will in turn determine whether a *continue* or a *quit* is issued.

Note here that these postconditions are entirely independent of the conversation class. The other agent need not be aware of what use is being made of the information, and only the receiving agent can be in a position to know what use is appropriate.

Completion conditions Policies and heuristics are less fruitful as a source of completion conditions, since when a performative is complete is something that must be agreed between the participants in the conversation. Therefore we should expect that these conditions derive always from the conversation class, and cannot, unlike preconditions and postconditions, be personal to an agent.

6 Discussion

It has emerged from the above that the concept of a conversation class is essential to permit communication between information sources of the sort envisaged in knowledge sharing applications. The notion of performatives is useful, but a performative needs to be given the context of a particular conversation class to be properly understood. In the context of different conversations a given performative may have different conditions associated with it. The conversation class tells us how a performative *can* be used, and sharing the conversation class harmonises the use of the performative across agents. In order for effective communication to be possible, the communicators must be operating within the same conversation class, so using the same conventions.

Additionally, however, agents will need to know how they *should* use the performatives. These conditions can be expressed in the same way as the conditions deriving from the the conversation class in a uniform framework of preconditions, postconditions and completion conditions. These conditions, however, need not be shared by both parties to the conversation. Since they do not affect the validity of the conversation, but rather the moves that will be chosen within it, the communicating agents are free to operate according to different policies and with different heuristics, as best befits their reasons for entering into the conversation.

We should not, therefore look for *the* specification of a performative. Rather what we need is to specify conversation classes. The specification of each conversation will comprise:

- A set of performatives which can be used in the conversations belonging to this class;
- For each such performative, a set of preconditions, postconditions and completion conditions regulating the use of that performative within the conversation class.

Two agents will be able to communicate if they have a conversation class in common. The conditions for the speech acts within a conversation class can then be augmented by additional conditions, which may be different for the two different agents, governing the moves they make in the particular conversation.

These conditions may also be different for a single agent in different instances of the conversation class. This gives a great deal of flexibility to the behaviour of agents within a conversation class.

An important feature of the method of specification given in this paper, is that conditions are expressed explicitly in terms of moves in the conversation rather than in terms of iterated cognitive modalities. This step towards an operationalisation may well be more appropriate to the kinds of information sources used in the application we took as a starting point, as it makes no commitment to the agents as full blown cognitive entities.

7 Conclusion

In this paper we have:

- Identified the need to specify autonomous communication between information sources in a generic manner;
- Given a framework for such specifications using performatives defined in terms of preconditions, postconditions and completion conditions.
- Distinguished between conditions which arise from the nature of the conversation, and which must be common to the communicating agents, and conditions which can be personal to the agents;
- Shown how we can replace iterated cognitive modalities by explicit reference to the conversation history.

We believe this framework provides a simple, and flexible but uniform, framework for the specification of communication between information sources in knowledge sharing applications.

References

1. Barbuceanu, M., and Fox, M.S., (1995), COOL: A Language for Communication in Multi Agent Systems in Lesser, V., (ed), Proceedings of the First International Conference on Multiagent Systems, MIT Press, Cambridge, Mass.
2. Bench-Capon, T.J.M., Argument in Artificial Intelligence and Law Artificial Intelligence and Law, 5(4), 249-61.
3. Finin, T., McKay, D., Fritzon, R., and McEntire, R., (1994) KQML: An Information and Knowledge Exchange Protocol in Fuchi, K., add Yokoi, T., (eds), Knowledge Building and Knowledge Sharing, Omsha and IOS Press.
4. Labrou, Y., (1996) Semantics for an Agent Communication Language, PhD Thesis, University of Maryland.
5. Neches,R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senatir, T., and Swartout, W., (1991) Enabling Technology for Knowledge Sharing, AI Magazine 12(3), pp36-56.
6. Wiederhold (1992) Mediators in the Architecture of Future Information Systems, IEEE Commuter, 25(3), pp38-49.
7. Wooldrige, M., and Jennings, N., (1995). Intelligent Agents: Theory and Practice, Knowledge Engineering Review 10(2).