Abstract Models for Dialogue Protocols

Raquel Fernández (raquel@ling.uni-potsdam.de)
Department of Linguistics, University of Potsdam
Karl-Liebknecht-Strasse 24-25, 14476 Golm, Germany
Tel: +49 331 977 2460, Fax: +49 331 977 2761

Ulle Endriss (ulle@illc.uva.nl)
ILLC, University of Amsterdam
Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands
Tel: +31 20 525 6511, Fax: +31 20 525 5206

Abstract. We examine a variety of dialogue protocols, taking inspiration from two fields: natural language dialogue modelling and multiagent systems. In communicative interaction, one can identify different features that may increase the complexity of the dialogue structure. This motivates a hierarchy of abstract models for protocols that takes as a starting point protocols based on deterministic finite automata. From there, we proceed by looking at particular examples that justify either an enrichment or a restriction of the initial model.

Keywords: Communication protocols, dialogue modelling, multiagent systems, theory of computation

1. Introduction

If we look at a corpus of real human-human dialogues, we find evidence of frequently reoccurring sequences of utterance types. For instance, questions are followed by answers and proposals are usually either accepted, rejected, or countered. These *interaction patterns* have inspired a line of research whose object of description is, broadly speaking, the rule-governed behaviour exhibited by dialogue interaction.

Interaction patterns are often modelled by means of *communication protocols*, i.e. public conventions that specify the range of possible follow-ups available to the participating agents. The focus of this paper is on the formal properties of such communication protocols. We examine a variety of protocols, taking inspiration from two fields: natural language dialogue modelling and multiagent systems. In the former case, protocols serve as an abstraction of dialogue structure, characterising what are the preferred continuations at a given point in a dialogue. In the case of multiagent systems, protocols are used to govern the interaction between autonomous software agents. Without first fixing an agent communication language and specifying a clear set of rules determining what an agent may "say" in a given situation, designing

agents that manage to communicate successfully would not be feasible in practice.

As we shall see, in dialogue interaction one can identify several features that have an impact on the complexity of the dialogue structure. In our approach, this variety of phenomena motivates a hierarchy of abstract models for protocols that is based on the expressive power of well-known machine models from the theory of computation. We do not claim that our classification subsumes the full range of protocols one can find in the literature; instead the hierarchy is intended as a classification that captures the relevant distinguishing features of different dialogue phenomena.

In the next section, we elaborate further on the notion of communication protocol. As the starting point of our hierarchy, in Section 3, we take protocols that can be modelled by *deterministic finite automata*. ¿From there, we proceed by looking at particular examples that justify either an enrichment or a restriction of the initial model. Our first example, at the beginning of Section 4, shows how to augment the basic model by a *stack* component to be able to represent protocols that can handle embedded dialogues. This kind of enrichment is then generalised to a class of *protocols with memory*. The subsequent sections analyse further instances of this class of protocols. After discussing the stack-based model in more detail in Section 5, we introduce protocols with a *stack of sets* in Section 6 to also account for compound moves within a single turn. Section 7 discusses protocols augmented with a simple *set*, which allow us to represent the kind of blackboard architecture used, for instance, in argumentation systems. The final example for our protocols with memory are the protocols equipped with a *list* presented in Section 8, which allow for an explicit representation of the dialogue history. A restriction of the basic automata-based model which allows for a simple logic-based representation of protocols is given in Section 9. Finally, our conclusions are presented in Section 10.

2. Communication Protocols

In communication modelling, it is common to distinguish between two main traditions: on the one hand, classical Artificial Intelligence approaches, inspired by ideas that originated in analytical philosophy, are built on general models of rational agency, emphasising the role that mental attitudes such as knowledge, belief, desire and intention play in conversational behaviour. This is the perspective adopted by plan-based approaches, most prototypically the BDI (Beliefs, Desires and Intentions) framework (see e.g. Cohen and Levesque (1990), Grosz

and Sidner (1990), or Sadek (1991)). On the other hand, one can identify a parallel line of research, that follows the work of philosophers like Lewis (1979) and Stalnaker (1978), and that instead of focusing on the intentional attitudes of the interacting agents and the plans that guide their contributions highlights the public and conventional aspects of communication. Under this perspective, a dialogue can be seen as a *conversational scoreboard* that keeps track of the state of the conversation.

A particular tendency within this latter tradition is the one inspired by the notion of *dialogue game* (Hamblin, 1970; Carlson, 1983) and the related concept of *adjacency pair* (Schegloff and Sacks, 1973; Levinson, 1983; Clark, 1996). The underlying idea here is that each participant's contribution determines a set of preferred options for follow-up in the dialogue. As stated already in the introduction, this relies on the evidence that conversations are composed of frequently reoccurring sequences of utterance types, such as questions being followed by answers and assertions being either acknowledged, discussed or elaborated upon. Communication protocols can then be seen as formal constructs modelling the public conventions behind these interaction patterns.

In multiagent systems, the use of conventional protocols has recently been put forward by a number of authors (Singh, 1998; Pitt and Mamdani, 1999a; Colombetti, 2000; Jones and Parent, 2004). This stands in marked contrast to the BDI or so-called mentalistic approach mentioned above, where the appropriateness or legality of a dialogue contribution is explained in terms of the mental attitudes of the agents participating in a dialogue (Cohen and Levesque, 1990; FIPA, 2002).

Conventional protocols have been shown to be a powerful descriptive and explanatory means of formalising the *rules of encounter* that characterise coherent interaction both in natural language dialogue, as well as in dialogue between autonomous software agents. In the case of multiagent systems involving autonomous software agents, protocols are simpler and typically more rigid, i.e. they describe the set of *allowed* or *legal* dialogue continuations. In natural language dialogue, on the other hand, protocols should be understood as characterising the range of *preferred* or *less-marked* follow-ups in particular dialogue situations. As such they do not restrict what counts as coherent in a general sense, but rather formalise in simple terms a range of (possibly ranked) unmarked follow-ups that reflects the expectations of the dialogue participants. In this sense, the violation of a protocol can also be informative, as it can be seen, for instance, as signalling a topic or task change. Thus, while in multiagent systems protocols are *prescriptive*

constructs, in natural language dialogue they tend to be *descriptive* and can therefore be evaluated in terms of their coverage of the data.[1]

It is worth pointing out that protocols need to be distinguished from *strategies*. While each dialogue participant may be equipped with their own strategy determining their actual responses, a protocol is a social concept common to all participants. That is, protocols are concerned with *shared* conventions, and as such can be thought of as part of the general dialogical competence of speakers. The notion of strategy on the other hand is a *private* one, and in this respect it is clear that a dialogue participant's strategy is shaped by the epistemic notions at the core of plan-based approaches. Protocols, in contrast, are not meant to determine what to say next, although of course they can be used to guide that decision process by specifying a range of possible next actions, or those actions with the highest probability.

For the dialogues that we consider in this paper, utterances are assumed to occur sequentially. This is a relatively common assumption in natural language dialogue modelling, whereas multiagent systems research has also tried to address concurrent communication. Also, we will mostly be concerned with dialogues involving two participants.[2]

3. Protocols as Finite Automata

Deterministic finite automata (DFAs) have been widely used to represent communication protocols, in particular in the area of multiagent systems (Parsons et al., 1998; Pitt and Mamdani, 1999b), although their use is also very common within the spoken natural language community (Bohlin et al., 1999). In this section, starting from a couple of simple examples, we introduce this class of DFA-based protocols and show how to define the central concept of *possible follow-up* with respect to this model. Most of the other protocol models discussed in subsequent sections are based on the basic model of DFA-based protocols.

[1] In this respect it is also worth mentioning statistical approaches, like e.g. Taylor et al. (1998) or Wright et al. (1999), that extract structural patterns from real data and then model protocols by assigning probabilities to the different options for follow-up. Although this is certainly an interesting perspective, the present approach abstracts from the possibility of statistically ranking allowed continuations.

[2] Some initial ideas on protocols for natural language dialogue involving multiple participants may be found in (Ginzburg and Fernández, 2005b).

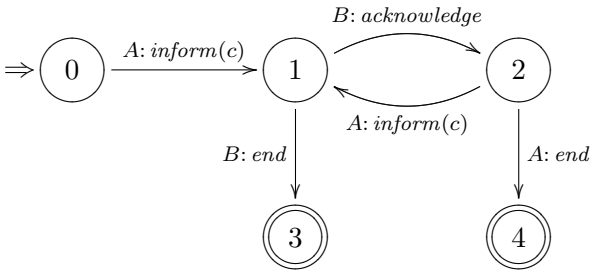

Figure 1. Continuous update protocol from (Pitt and Mamdani, 1999b)

3.1. Some Examples

Pitt and Mamdani (1999b) give several examples for automata-based protocols. One of them, the *continuous update protocol*, specifies a class of dialogues between two agents A and B where A continuously updates B on the value of some proposition. Figure 1 provides an intuitive description of this protocol. Here, c is an expression in some suitable content language which is used to encode the actual information transmitted by A. For this particular protocol, the value of c is intended not to be relevant; any *inform* move uttered by agent A will take us from state 0 to state 1, whatever the content of the transmitted piece of information may be.

The notion of what constitutes a legal dialogue conforming to the above protocol is intuitively clear. At the time a new dialogue starts, for instance, an *inform* move uttered by agent A would be the only legal utterance. Immediately after A has *inform*ed B, the latter can either choose to *acknowledge* that fact or it may decide to *end* the dialogue. However, it would be illegal for A to continue the dialogue with another *inform* move unless it has received an *acknowledge*ment from B first, and so forth.

Several spoken dialogue systems also use DFA-based models to determine the course of well-formed conversations. One of them is the SRI-Autoroute system (Lewin, 1998), which is based on Conversational Game Theory (Power, 1979). In this system finite automata are used to model *games* that represent the conversational rules governing exchanges. Figure 2 shows a graphical representation of a *confirmation game* given by Lewin (1998). This automaton characterises what the system (A) can expect from the user (B) in an interaction where A asks B for confirmation of some utterance. In this situation the user is expected to reply either affirmatively with a *reply_yes* move or negatively with a *reply_no* move. Alternatively, the user may correct

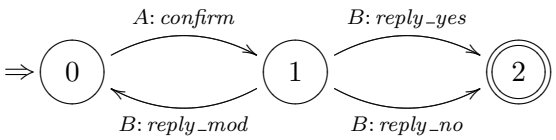

Figure 2. Confirmation game from (Lewin, 1998)

the hypothesis of the system by a *reply_mod* move, in which case the confirmation game starts again.

An example for a more sophisticated DFA-based protocol would be the finite state model of grounding proposed by Traum (1994). This protocol provides constraints on possible grounding act sequences, allowing for a fine-grained monitoring of the grounding status of an utterance.

3.2. DFA-based Protocols

The type of protocols discussed in our examples above will provide the starting point for our proposed classification of communication protocols. We are now going to define the class of *DFA-based protocols*, i.e. the class of protocols that can be defined in terms of a DFA. Our definition amounts to a simple re-wording of the usual definition of a DFA (Hopcroft et al., 2001; Lewis and Papadimitriou, 1998) using a terminology appropriate for the description of dialogue protocols.

DEFINITION 1 (DFA-based protocol). *A DFA-based protocol is a quintuple $\langle Q, q_0, F, \mathcal{L}, \delta \rangle$, consisting of a finite set of dialogue states Q, including an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$, a communication language $\mathcal{L}$, and a transition function $\delta : Q \times \mathcal{L} \to Q$.*

The elements of the communication language $\mathcal{L}$ are *utterances* and are constructed from a finite set $\mathcal{A}$ of *agents* (or *dialogue participants*), a finite set $\mathcal{M}$ of *dialogue moves* (or *illocutionary acts*), and a *content language* $\mathcal{C}$. We assume that every utterance has the structure $i : m(c)$ with $i \in \mathcal{A}$, $m \in \mathcal{M}$, and $c \in \mathcal{C}$. In general, at the level of describing abstract models for dialogue protocols rather than concrete instances of these models, we do not put any restrictions on the content language $\mathcal{C}$, i.e. utterances of the form $i : m(c)$ cover any type of utterance. Our chosen representation merely singles out the name of the speaker and the type of the dialogue move. As the types of dialogues we are going to consider typically only involve two participants we may think of $\mathcal{A}$ as the set $\{A, B\}$.

When talking about DFAs in general (i.e. not just in the context of protocols), we would refer to $\mathcal{L}$ as an *input alphabet* rather than a communication language. The input alphabet is usually defined as a finite set (Hopcroft et al., 2001). This may seem at odds with our set of utterances $\mathcal{L}$ which, intuitively (at least in the context of natural language), could be infinite. While this may indeed be the case, we can always group utterances into a finite number of equivalence classes with respect to the effect they have on the state transition function δ. For any given input state, there are only a finite number of output states the system could move to. Typically, in the context of communication protocols, these equivalence classes are determined by the dialogue moves in $\mathcal{M}$, which is a finite set.

Representing protocols as DFAs allows for a simple formalisation of the notion of *possible follow-up* at a given point in a dialogue:

DEFINITION 2 (Possible follow-up). *Given the current dialogue state q, an utterance u constitutes a possible follow-up of the dialogue iff there exists a state $q' \in Q$ such that $\delta(q, u) = q'$ holds.*

Before a dialogue starts we are in the initial state q_0. The dialogue state then gets updated whenever an utterance is performed, following the transition function δ. A complete dialogue *conforms* to a given protocol iff it is *accepted* by the DFA, i.e. iff each utterance in the dialogue is a possible follow-up and the final utterance leads to a final state in F.

In the context of multiagent systems, where protocols have a prescriptive function, possible follow-ups may also be interpreted as *legal* follow-ups. We will use the latter term when appropriate.

4. Protocols with Memory

In this section, we are going to see a first example for a dialogue feature that cannot be modelled by a simple DFA-based protocol in a satisfactory manner. As we shall see, this observation gives rise to an extension of our basic model, which involves adding a *memory component*. This component allows us to store utterances (or abstractions thereof) that may affect the range of possible follow-ups later on in the dialogue.

This section describes the extended model in general, while subsequent sections discuss specific instances of the general model and their applications in detail.

4.1. Protocols that Allow for Subdialogues

In natural language dialogue it is not uncommon to find embedded pairs of questions and answers, where a sequence of questions is followed by a sequence of answers, which answer the questions in reverse order. This is exemplified in the following dialogue, taken from Levinson (1983):

(1) A: May I have a bottle of Mich?	$[Q_1]$
(2) B: Are you twenty one?	$[Q_2]$
(3) A: No.	$[A_2]$
(4) B: No.	$[A_1]$

The next example shows that deeper embeddings are also possible:

(1) A: Who should we invite?	$[Q_1]$
(2) B: Should we invite Bill?	$[Q_2]$
(3) A: Which Bill?	$[Q_3]$
(4) B: Jack's brother.	$[A_3]$
(5) A: Oh, yes.	$[A_2]$
(6) B: OK, then we should invite Gill as well.	$[A_1]$

Replying to a question with another question (2) and asking for clarification (3) are very common phenomena in natural language dialogue, especially in information-oriented interaction. A protocol characterising this kind of dialogues would, at the very least, have to be able to keep track of the number of questions asked so that the number of answers can be matched against it. If the number of questions is not bounded, this would require an unlimited amount of memory to be able to store that number. This is not possible with DFA-based protocols, because DFAs have a limited amount of memory, encoded by the fixed set of states of the automaton.

Thus, the presence of embedded subdialogues (or *insertion sequences* in the terminology of Levinson (1983)) creates a structure that calls for an enrichment of the DFA-based model. This can be modelled by adding a *stack* to a DFA. In the example above, questions would get pushed onto the stack, to be then popped by their respective answers. We are going to discuss this abstract model for dialogue protocols in detail in Section 5. As is well-known, the machine model of a DFA together with a stack corresponds to a *pushdown automaton* (Hopcroft et al., 2001; Lewis and Papadimitriou, 1998).

4.2. Adding a Memory Component to the Basic Model

Storing questions in the manner suggested above is an example for an *abstraction* from the full dialogue history. We only keep those parts of the history that are relevant to the choice of future follow-ups, and we do so in a convenient format. For embedded question-answer sequences, an appropriate format seems to be that of a stack.

DFAs are abstract machines with a limited amount of memory. Adding a (finite) stack amounts to enriching the automaton with an unlimited memory component. Modelling this memory as a stack is just one of many options. Besides stacks, we may consider a variety of *abstract data types* (ADTs) such as, for instance, *queues*, *sets* or *lists* (Aho et al., 1983).[3] We call the set of objects that can be stored in memory the *memory alphabet* (which may or may not be identical to the communication language). Every ADT comes with a set of basic operations (*push*(x) and *pop* in the case of a stack) and functions (*top* to return the top element on a stack, for example). A *configuration* of a memory component is an instance of the ADT used to model that memory component. For example, in the case of a set, a configuration would be a subset of the memory alphabet, while for a stack it would be a string of elements of that alphabet. The *visible* part of a configuration is the part that can be checked using the available ADT functions. In the case of a stack, for instance, only the topmost element is visible. In the case of a set, on the other hand, all elements are visible.

For any given ADT, we can define a class of *protocols with memory* based on that ADT as follows:

DEFINITION 3 (Protocol with memory). *A protocol with memory based on a given ADT is a sextuple* $\langle Q, q_0, F, \mathcal{L}, \mathcal{L}', \delta \rangle$*, consisting of a finite set of dialogue states* Q*, including an initial state* q_0 *and a set of final states* $F \subseteq Q$*, a communication language* $\mathcal{L}$*, a memory alphabet* $\mathcal{L}'$*, and a transition function* $\delta : Q \times \Gamma \times \mathcal{L} \rightarrow Q \times \Gamma$*, where* Γ *denotes the set of all possible configurations of the memory component.*

There are two restrictions on δ. Firstly, the definition of δ may only refer to the visible part of the input configuration, and it has to be implementable in terms of the available ADT operations. Secondly, δ has to be representable in terms of a finite subset of $(Q \times \Gamma \times \mathcal{L}) \times (Q \times$

[3] The use of ADTs plays an important role in the definition of the information state in implemented dialogue systems following the information-state approach to dialogue modelling, developed in the TRINDI project (Larsson and Traum, 2000). For a list of some ADTs useful for dialogue management in such computational systems see e.g. Traum et al. (1999), Larsson (2002), and Bos et al. (2003).

Γ).[4] The latter means that, as for the communication language, even if the memory alphabet may be infinite to begin with, from an abstract point of view, we only need to distinguish a finite number of equivalence classes of elements of the alphabet when modelling the automaton.

Our definition of what constitutes a possible follow-up at a given point during a dialogue for this extended model is very similar to the case of DFA-based protocols:

DEFINITION 4 (Possible follow-up). *Given the current dialogue state q and the current configuration of the memory component x, an utterance u constitutes a possible follow-up of the dialogue iff there exist a state $q' \in Q$ and a configuration $x' \in \Gamma$ such that $\delta(q, x, u) = (q', x')$.*

As before, a complete dialogue *conforms* to a given protocol with memory iff it is accepted by the automaton in question, i.e. iff each and every utterance is a possible follow-up and the last utterance takes us to one of the final states in F.

In the following sections, we are going to discuss several choices for ADTs as memory components enriching the basic DFA-based model, which are required to account for different dialogue phenomena.

5. Protocols with a Stack

As we have seen at the beginning of Section 4, not all dialogue structures are satisfactorily captured by a protocol with a machine model that corresponds to a simple DFA. In particular, we have argued that the phenomenon of embedded subdialogues can be modelled by adding a finite *stack* to a DFA-based protocol. A stack allows us to store arbitrarily large amounts of information that are accessible in a last-in-first-out (LIFO) manner. Such information can be manipulated by means of the function *top*, which returns the top element on the stack, and the operations *push*(x), which pushes element x onto the stack, and *pop*, which removes the top element from the stack. In this section, we are going to discuss the model of a DFA-based protocol enriched with a stack component in more detail and give a couple of examples for relevant dialogue phenomena.

[4] These restrictions are in line with the standard definition of a pushdown automaton (Lewis and Papadimitriou, 1998), which is the prototypical example of an automaton with an explicit memory component.

5.1. Questions Under Discussion and More

An example of a structuring mechanism able to handle embedded question-answer sequences is Ginzburg's QUD (*questions under discussion*) (Ginzburg, 1996; Ginzburg, 2006). This is a storage device that nicely accounts for the fact that, although several issues can be relevant at a particular stage in a dialogue, they are usually ranked in terms of relative salience. Simplifying a little, in Ginzburg's approach questions get introduced into QUD and get *downdated* once they are answered. The assertion of a proposition p also introduces a question into QUD, namely *whether*(p). Typically, this question would get downdated from QUD once p is acknowledged.[5]

In Ginzburg's theory, the QUD plays a central role in determining the possible responses to a given utterance, the general assumption being that the turn-holder will address the top element in QUD. In the context of embedded question-answer sequences, the last question asked becomes the most salient question under discussion, which will be the first one to be answered. Thus, a stack seems the appropriate ADT to characterise Ginzburg's QUD. Several dialogue systems, like for instance GoDiS (Larsson et al., 2000; Larsson, 2002), have indeed implemented the QUD model as a stack. We are going to discuss Ginzburg's framework further in Section 6.

Besides *questions under discussion*, other aspects relevant to dialogue management can be successfully modelled with a stack. A clear example are *discourse obligations* (Traum and Allen, 1994; Matheson et al., 2000), which constitute a structuring mechanism similar to QUD. The main idea of the approach is that some utterance types impose obligations on the addressee to respond to these utterances. The assumption is then that dialogue participants will always try to address the topmost element on their obligation stack. Another notion that can be (and indeed usually is) modelled by means of a stack is the *agenda of actions* to be performed by an agent (Traum et al., 1999). Again, the assumption is that an agent will always try to perform the action on top of its agenda. In general terms, thus, any repository of information that requires only limited accessibility related to recency—corresponding to the LIFO property of stacks—can in principle be modelled by this kind of datatype.

[5] A protocol for inquiry-oriented dialogues based on this framework has also been formalised using first-order dynamic logic (Fernández, 2003). This suggests, more generally, that it is possible to specify our protocols, including the ADT operations on the memory component, in logical terms.

5.2. Expressive Power

As pointed out earlier already, a DFA together with a stack corresponds to a *pushdown automaton*. As is well known, pushdown automata are strictly more powerful than DFAs (Lewis and Papadimitriou, 1998). In our context this means the following:

FACT 1. *The class of dialogues conforming to protocols with a stack strictly includes the class of dialogues conforming to DFA-based protocols.*

It may also be interesting to use protocols with more than one stack. Kreutel and Matheson (1999) and Traum (2003), for instance, propose to use a stack of obligations together with a stack of questions under discussion. It is important to note though that adding a second stack to a protocol with a stack would (again) increase expressive power, because a pushdown automaton with two stacks is equivalent to a Turing machine, which is strictly more powerful than a simple pushdown automaton (Lewis and Papadimitriou, 1998).

While these observations regarding the expressive power of different protocol models are easy exercises from a computation-theoretic point of view, we do believe that they offer an interesting and novel perspective on dialogue modelling. The type of abstract machine that is (usually implicitly) encoded when specifying a particular agent interaction protocol or dialogue management system is certainly *one* relevant dimension according to which we can classify the complexity of such a system. Of course, it is also clear that not every protocol with, say, two stacks will fully exploit the power of Turing machines. Indeed, a "simple" Turing machine may be far less complex that a "complicated" pushdown automaton.

6. Protocols with a Stack of Sets

In Section 4, we have considered a dialogue where several questions are posed in sequence. There we have argued that in these cases it seems reasonable to use a protocol where the last question asked takes precedence (i.e. it is the first one to be addressed), and that such a protocol can be modelled by a pushdown automaton. In this section, we are going to discuss a generalisation of this model, which makes accessibility more flexible.

6.1. Successive Queries by the same Speaker

As some authors have noticed (Asher, 1998; Ginzburg, 2006), when successive queries are asked by a single speaker, the simple kind of protocol discussed in the previous section would not always account correctly for the data. This is illustrated by the following example (adapted from Asher (1998)):

(1) A: Where were you on the 15th?	$[Q_1]$
(2) A: Do you remember talking to anyone after the incident?	$[Q_2]$
(3) B: I didn't talk to anyone.	$[A_2]$
(4) B: I was at home.	$[A_1]$
(3') B: I was at home.	$[A_1]$
(4') B: I didn't talk to anyone.	$[A_2]$

Dialogues such as the one above show that when two or more questions are uttered in succession by the same speaker, the respondent is sometimes allowed to answer them in any order. In such dialogues, the questions under discussion are in what has been called a *coordinate structure* (Asher, 1998), with none of them taking precedence over the others. When this is the case, a protocol based on a DFA plus a stack would not be appropriate to handle this phenomenon.

Larsson (2002) gives some examples that point in the same direction. As he notes, assuming that the questions in the following dialogue are organised as a stack "suggests a very unintuitive interpretation of B's answer, where 10:30 is the time when B is coming back and 11:30 is the time when B is leaving".

(1) A: When are you leaving?	$[Q_1]$
(2) A: When are you coming back?	$[Q_2]$
(3) B: Ten thirty and eleven thirty.	$[A_?]$ & $[A_?]$

In fact, although in Section 5 we have proposed to represent the QUD as a stack, in Ginzburg's model the questions currently under discussion form a *partially ordered set*. This order indicates what conversational precedence different questions take over each other. It should be noted, however, that the proposed model does not actually make use of the full expressive power of a partially ordered set. This is so because new questions can only be added at the top, either strictly above the currently most salient question, or next to it. Also, questions can only be

deleted from the top; we do not have access to elements further down the order (Ginzburg, 2006).

In terms of our hierarchy of protocols with memory, such an architecture can be modelled using a DFA together with a finite *stack of sets*. The questions under discussion that currently have maximal conversational precedence are those in the top set of the stack. Now, adding a new question strictly above the currently maximal ones corresponds to pushing a singular set containing only that question onto the stack. To add a new question *next* to the currently maximal questions, on the other hand, we first pop the top set off the stack, then insert the new question into that set, and then push the new set back onto the stack. To delete a question (with maximal precedence), we first pop the top set off the stack, then delete that question from the set, and finally push the remaining set back onto the stack—unless that set is empty (i.e. unless there has been only a single maximal question). A delete operation will fail in case the question given as a parameter is not a member of the top set.

Larsson (2002) proposes to model QUD as an *open-stack*, i.e. a structure that retains the stack order but possesses set-like properties. For instance, as a set, an open stack cannot contain repeated elements. This makes $push(x)$ a complex operation that involves first checking whether x is a member of the open-stack; in the affirmative case, x is deleted prior to being pushed on the top position. Even though non-topmost elements can be accessed and deleted, questions can only be added at the top. Contrary to our *stack of sets*, however, this model cannot account for bunches of elements that are "at the same level" while being ordered with respect to other elements. As will be shown (for sets) in Section 7, the expressive power of the *open-stack* model is not higher than that of a DFA.

6.2. Choosing the Right Operation: Push or Insert?

An interesting issue is what causes a question to be inserted into the top set of the stack or, alternatively, to be pushed onto the currently maximal set. A simple hypothesis we could make is that the first operation takes place when successive queries are posed within a single turn (as in the examples above), while the second one is executed when a different speaker replies to a question with another question (as in the examples in Section 4). The following dialogue, taken from Ginzburg (2006), however, suggests that successive querying within a single turn does not always imply that the questions enjoy equal status:

(1) A: Who will you be inviting?	$[Q_1]$
(2) A: And why?	$[Q_2]$
(3) B: Mary and Bill, I guess.	$[A_1]$
(4) A: Aha.	$[Ack]$
(5) B: Yeah, (because) they are very undemanding folks.	$[A_2]$

Notice that here the first question asked seems to take precedence over the last one—only after the first question is answered does the second question get addressed. Although a reply along the lines of *"I'd like to have a quiet dinner, so Mary and Bill I guess"* would also be possible, the order of answers in the dialogue above seems to be the least marked option in this situation.[6] This suggests that the order in QUD is not determined solely by conventional means (i.e. in terms of the order of occurrence in the dialogue or the speakers of the questions), but is also guided by semantic relations that may hold between its elements. This is of course noted by Asher (1998) and Ginzburg (2006), who explicate the differences in terms of the relations that are said to hold between the questions: *coordination* in case questions can be answered in any order, and *query-extension* in case they are more naturally answered in the order in which they have been posed.[7]

There is not much to say about this from the abstract point of view we take here, besides noting that complex relations between the elements of the content language $\mathcal{C}$ would have to be encoded as part of the definition of the transition function δ.

6.3. Expressive Power

It turns out that, from a computational point of view, the model of a DFA enriched with a stack of sets is in fact not more expressive than that of a pushdown automaton (with a simple stack):

FACT 2. *The class of dialogues conforming to protocols with a stack is the same as the class of dialogues conforming to protocols with a stack of sets.*

[6] In early work, Ginzburg uses examples like this to motivate a modification of his QUD-*update* protocol, namely the addition of a new operation ("$+_{\text{QUD-FLIP}}$"), which pushes a question *under* the maximal element in QUD, i.e. between the topmost element and the rest of the stack. In the abstract model proposed here this would correspond to first popping the top element off the stack, then pushing the new question, and finally pushing the former top element back onto the stack.

[7] Since the early work of Mann and Thomson (1987), a lot of attention has been paid to characterising the nature of *discourse relations*, i.e. coherence relations that hold between adjacent sentences, mostly in text/monologue (see e.g. Asher (1993)), but more recently also in dialogue (Asher and Lascarides, 1998).

This may be seen by considering that, given a DFA with a stack of sets using the memory alphabet $\mathcal{L}'$, we can construct a pushdown automaton that uses the power-set of $\mathcal{L}'$ as its memory alphabet and that accepts the same inputs as the original automaton.

An alternative way of constructing a pushdown automaton that is equivalent to a given DFA with a stack of sets would be to extend the memory alphabet by a special "separator" symbol. Now all elements in the stack between two separators would be considered a set. Each state in the original DFA would have to be replaced with a sub-automaton to simulate checking for membership and deleting elements from the top set.

In either case, our simulation would result in an exponential blow-up of the model, either with respect to the memory alphabet or with respect to the number of states. Therefore, in practice, the model of a DFA-based protocol with a stack of sets may often be preferred over protocols with a simple stack.

7. Protocols with a Set

So far we have seen abstract models for protocols that are related to different degrees of recency, namely protocols that make use of a stack or a stack of sets. Sometimes, however, determining what constitutes an appropriate follow-up with respect to a protocol can be related to aspects that are independent from the order in which inputs have been received. To account for this, as a further instance of the general model of protocols with memory, we are going to discuss DFA-based protocols extended with a *set*.

The ADT operations available when working with a memory component structured as a set are *insert*(x) to insert element x into the set and *delete*(x) to remove it again. The central function available is *member*(x) to test whether x can be found in the set (Aho et al., 1983).

7.1. The Blackboard Architecture

An example of an order-independent model is the so-called *blackboard architecture*, which has been used in the context of multiagent systems to formalise protocols inspired by work on argumentation in dialogue modelling. Argumentation-based protocols have been used to model different types of dialogues (such as negotiation dialogues or persuasion dialogues) between software agents (Amgoud et al., 2000). Central to this approach is the notion of a *commitment store*, due to Hamblin (1970).

For example, *asserting* a proposition amounts to placing that proposition into one's commitment store. A *retract* move would then be considered legal only if the corresponding proposition can be found in the agent's commitment store (and would itself cause the respective proposition to be deleted again). Similarly, to model the rule that an agent may only challenge a proposition that has previously been asserted by its opponent, we may stipulate that a *challenge* move is only legal in a situation where the proposition that is being challenged has previously been added to the commitment store.

This kind of blackboard architecture may be modelled in terms of a DFA-based protocol together with a finite *set* (or possibly one set for each agent). Any utterances that may affect the legality of utterances later on in a dialogue would be stored in this set. In particular, this kind of architecture requires us to abstract from the order in which utterances occur. We can only keep track of the fact that a given utterance either has or has not been uttered in the past.

Besides modelling argumentation-based dialogue, another application of this model would be agent interaction protocols involving *social commitments* (Singh, 1998; Colombetti, 2000). An example, taken from Singh (1998), would be that an agent, if asked for a price quote by different agents, must always reply with the same quote. This social commitment may be modelled by storing the first price quote in the set component (possibly together with the query and the time of the first query). Any subsequent reply to a price quote may then be checked against the contents of the set to detect potential violations of the protocol.

A final example for the use of a set component in a protocol is the set of FACTS, which is a further component of the information state as modelled in Ginzburg's theory (besides QUD). This is a set the elements of which do not represent the commitments or beliefs of agents, but rather what has been established for the sake of the conversation (Ginzburg, 1996). One may want to design protocols where, for instance, questions are only appropriate follow-ups whenever their answers are not members of the set of FACTS.

7.2. Expressive Power

It is interesting to note that adding a set to a DFA does in fact not increase expressive power, because the range of all possible configurations of the set component could be encoded into a larger DFA:

FACT 3. *The class of dialogues conforming to DFA-based protocols is the same as the class of dialogues conforming to protocols with a set.*

This is the case, because we are working with a *finite* memory alphabet.[8] The set of possible configurations of the blackboard is the power-set of the memory alphabet, i.e. it is also finite. We can therefore transform the original DFA by introducing a new state for every pair of an original state and a configuration of the blackboard. If we arrange the transition function accordingly, we obtain a new DFA (without explicit memory component) that corresponds to the same protocol as the original automaton. However, such a construction would result in an exponential blow-up of the set of states; that is, in practice, a blackboard architecture can have great advantages over a simple DFA-based protocol.

In the literature on argumentation systems, each agent is usually equipped with its own commitment store. Again, while this is a convenient means of representation, working with a DFA with more than one set does also not increase the expressive power of the model, because the range of all possible configurations of the memory components could be encoded explicitly within a larger DFA.

8. Protocols with a List

Besides stacks and sets, another important ADT is the *list* data type. Like a stack, a list can be used to store a string of elements of the memory alphabet, but, while a stack only allows access to its top element, in a list all elements are visible and can be manipulated. The most important ADT operations for lists are $\mathit{insert}(i, x)$ to insert element x at position i and $\mathit{delete}(i)$ to remove the element stored at position i from memory, while the function $\mathit{retrieve}(i)$ can be used to check the value of that element (Aho et al., 1983).

In this section, we briefly discuss possible applications for protocols based on DFAs enriched with a list component and comment on the expressive power of this abstract model.

8.1. Explicit Representation of the Dialogue History

Systems providing access to (parts of) the dialogue history explicitly in order to check the legality of an utterance may be modelled as DFA-based protocols together with a finite *list* (by appending utterances to the end of the list as they occur in the dialogue). This architecture

[8] As discussed in Section 4, even if the memory alphabet, e.g. the set of all possible utterances, is assumed to be infinite, it can be reduced to a finite number of equivalence classes with respect to the effect an utterance has on the course of the dialogue.

allows us to keep track of relevant utterances and the order in which they occur. In particular, a list-based representation enables us to access any of the elements stored in memory at any time, and not just, say, the element inserted last (as for stacks).

An architecture of this sort has been used by Ginzburg and Fernández (2005a) to account for acceptance moves in multi-party dialogue, where the asserter of a proposition p can consider p accepted by the audience only when it has been accepted by *all* addressees. This is modelled by means of a list of MOVES that keeps track of the utterances that occur in the dialogue. A proposition p is considered accepted if MOVES contains an $assert(p)$ move followed by acceptance moves by all addressees participating in the dialogue. Thus we need to access several moves stored in memory, and to make sure that the assertion occurs prior to the acceptances we also need to keep track of the order in which moves occur.

Protocols with a list are the most powerful protocol model we have discussed, because they allow us to record the full dialogue history, which—in principle—should always make it possible to specify *any* conditions pertaining to the legality of an utterance. In fact, this is precisely the thesis underlying the conventionalist approach to communication protocols (in multiagent systems research): what is legal may only depend on publicly observable facts (Singh, 1998).

Of course, in computational terms, this model is also the most costly one. Storing the entire dialogue history may not always be feasible. Also, simply storing the history without making suitable abstractions (as in our previous examples) will often be too rich a mechanism for designing concise protocols.

8.2. Expressive Power

The machine model of a DFA enriched with a finite list is equivalent to a *Turing machine* (Hopcroft et al., 2001; Lewis and Papadimitriou, 1998). This follows immediately from the fact that the list can be used to store the tape, while the DFA can be used to implement the control component of a Turing machine. An alternative way of showing this equivalence would be to build on the fact that a list can be used to encode two stacks and, as mentioned earlier already, a pushdown automaton with two stacks is equivalent to a Turing machine. It follows that protocols with a list are strictly more expressive than protocols with a stack (and thereby also than simple DFA-based protocols):

FACT 4. *The class of dialogues conforming to protocols with a list strictly includes the class of dialogues conforming to protocols with a stack.*

As a final remark, it is well known that Turing machines with multiple tapes can be simulated by a single-tape Turing machine (Lewis and Papadimitriou, 1998). Therefore, working with a protocol with several list components would not increase expressive power any further.

9. Shallow Protocols

So far we have concentrated on enriching the basic model of DFA-based protocols to cater for a variety of complex dialogue phenomena. Where such phenomena are not present, we may usefully restrict the model rather than extend it.

Recently, a class of so-called *shallow protocols* has been introduced in the context of multiagent systems (Endriss et al., 2003). A shallow protocol is a protocol where the legality of an utterance can be determined on the sole basis of the previous utterance in the dialogue.

9.1. Shallow Rules and Conformance Checking

For example, to express that any proposal by agent A must be followed by either an acceptance, a rejection, or a counter proposal by agent B, we may use the following shallow rule (omitting descriptions of the content of an utterance for simplicity):

$$A\text{: } propose \;\rightarrow\; \circ\,(B\text{: } accept \;\vee\; B\text{: } reject \;\vee\; B\text{: } counter)$$

Here we use the *next-operator* $\circ$, familiar from linear temporal logic (Goldblatt, 1992), to refer to the next turn in the dialogue.[9]

In general, a shallow rule is of the form $u \rightarrow \circ(u_1 \vee \cdots \vee u_n)$, where u as well as $u_1, \ldots, u_n$ are utterances. This rule expresses that any of the utterances on the righthand side constitutes a legal continuation of a dialogue where the latest utterance has been u. In addition, we require a special symbol to indicate the start of a dialogue (to be able to provide a rule of the above form specifying the range of legal initial utterances). There are a number of requirements a set of shallow rules forming a protocol need to meet. Notably, there can only be (at most) one rule for any "trigger" u. For full details we refer to Endriss et al. (2003).

While such an approach to representing protocols may seem somewhat simplistic, it nevertheless applies to a wide range of protocols proposed in the literature (some of which are mentioned below). Specifically, it can be of particular interest in the area of multiagent systems.

[9] Refer to Endriss (2005) for a discussion of the representation of dialogue protocols using temporal logic.

As shown by Endriss et al. (2003), it is possible to check *a priori* whether a given agent will behave in conformance to a given shallow protocol by inspecting the agent's specification, rather than just observing an actual dialogue at runtime. This is the case, at least, in the sense of the agent being guaranteed not to utter anything illegal; guaranteeing that an agent actually utters anything at all turns out to be somewhat more difficult a problem. The ability to check conformance to a protocol not only at runtime, i.e. while a dialogue is actually taking place, but also before entering into an interaction with other agents, allows system designers to build more successful software agents which, for example, may be able to avoid penalties associated with behaviour that is deemed illegal according to the social interaction protocol in place.

9.2. Shallow Protocols as Automata

Shallow protocols may be understood either in terms of reactive rules of the kind given above or as restricted DFA-based protocols.

DEFINITION 5 (Shallow protocol). *A DFA-based protocol is shallow iff the value of the transition function δ is always uniquely identifiable given only its second argument (the utterance).*

Thus in a shallow protocol, to determine the state we will end up in given an utterance u, we do not need to know the state we come from. A shallow rule tells us that, regardless of that state, whenever utterance u on the lefthand side of the rule takes place, we end up in a state whose possible outgoing utterances are those given on the righthand side of the rule. The protocol shown in Figure 2 representing the confirmation game proposed by Lewin (1998), for instance, is a clear example for a shallow protocol, because each utterance is only used to label a single transition.

A *non*-shallow protocol would be a DFA where two edges with the same label point to two different states. For example, it is common for spoken dialogue systems to include a protocol that allows the system to ask the user to repeat their input at most, say, three times in case of recognition problems. If the problem persists after the third repetition, the system would terminate the interaction and possibly pass on to a human operator. This change of behaviour after the third repetition cannot be modelled by means of a shallow protocol.

In principle, it is always possible to turn a DFA-based protocol into a shallow protocol by renaming any duplicate transitions. In fact, many of the simpler DFA-based protocols to be found in the multiagent systems literature happen to be shallow or could at least be made

shallow by renaming only a very small number of transitions (besides Lewin (1998), examples include the protocols proposed by Parsons et al. (1998) and Pitt and Mamdani (1999b)). In such cases, it seems advantageous to use the simpler model of shallow protocols in the first place.

9.3. Expressive Power

In a context where the renaming of utterances labelling problematic transitions is not acceptable, however, shallow protocols really do determine a proper subclass to DFA-based protocols.

FACT 5. *The class of dialogues conforming to DFA-based protocols strictly includes the class of dialogues conforming to shallow protocols.*

While there is no standard machine model that corresponds to the class of shallow protocols, we can nevertheless characterise it in terms of the type of grammar that could generate a dialogue following a shallow protocol. Recall that the regular languages, which are the languages accepted by DFAs, are the languages generated by *right-linear grammars* (Lewis and Papadimitriou, 1998). A context-free grammar is called right-linear iff the righthand side of every rule in the grammar consists of any number of terminal symbols followed by at most one non-terminal. In fact, for any regular language there exists a right-linear grammar generating that language where there is also at most one terminal symbol in any rule. To characterise shallow protocols, we have to impose an additional restriction on the structure of admissible grammar rules.

DEFINITION 6 (Shallow-right-linear grammar). *A context-free grammar is shallow-right-linear iff* (*i*) *every rule is of the form* $A \to bB$ *or* $A \to b$*, where* A *and* B *are non-terminal symbols and* b *is a terminal symbol; and* (*ii*) *for every terminal symbol* b *there is a unique context in which it may appear: either* b *only appears on its own or it only appears with a unique non-terminal symbol* B *(that is, as* bB*).*

The first part of this definition characterises right-linear grammars, while the second part encapsulates the shallowness condition. Now the languages generated by shallow-right-linear grammars over the alphabet given by a communication language $\mathcal{L}$ are precisely the strings of utterances that correspond to legal dialogues according to shallow protocols.

Table I. Abstract models of dialogue protocols

Abstract model	*Examples*
Shallow rules ($\subset$ DFA)	– simple communication protocols in multiagent systems (Endriss et al., 2003)
Finite automaton (DFA)	– simple communication protocols in multiagent systems (Pitt and Mamdani, 1999b) – conversational games (Lewin, 1998) – finite-state model of grounding (Traum, 1994)
DFA + set (= DFA)	– commitment store in argumentation (Hamblin, 1970; Amgoud et al., 2000) – social commitments in multiagent systems (Singh, 1998; Colombetti, 2000) – commonly accepted FACTS (Ginzburg, 1996)
DFA + stack (= pushdown automaton)	– (simplified) questions under discussion (Ginzburg, 1996) – (some models of) discourse obligations (Kreutel and Matheson, 1999)
DFA + stack of sets (= pushdown automaton)	– questions under discussion (Ginzburg, 1996)
DFA + list (= Turing machine)	– explicit representation of dialogue history – MOVES in multi-party dialogue (Ginzburg and Fernández, 2005a)

10. Conclusion

In this paper, we have analysed a variety of interesting features of dialogue as they occur either in natural language interaction or in the context of multiagent systems. These features have given rise to a number of different abstract models for dialogue protocols. These protocols are based on the machine model of a deterministic finite automaton, which we have further enriched with memory components modelled as different abstract data types. In one case, we have also seen that a restriction of the basic model can have useful applications. Table I gives an overview of the various protocol models we have discussed, together with a selection of representative examples.

We should emphasise that our protocols with memory are *abstract* models that are intended to capture characteristic features of particular classes of dialogues. In most cases, the full power of the theoretical

model will not be necessary to account for actual human-human dialogues. For instance, pushdown automata, which we have used to model the phenomenon of subdialogues, are only strictly more expressive than simple DFAs if the size of the stack is not bounded. However, one may argue that humans will hardly ever use more than a relatively small number of levels of embeddings. In the case of communicating software agents this bound may be higher, but for practical purposes it seems still very reasonable to work with an upper bound on the number of elements that can be stored in the stack. For all the ADTs that we have discussed in this paper, if the number of elements that can be stored is bounded, then a DFA equipped with a respective memory component is no more expressive than a simple DFA. We stress that this does not disqualify the idea of working with memory-enriched protocols, however. A simple DFA together with an ADT that structures relevant information in an appropriate manner can be much more useful, from both a practical and a theoretical point of view, than a single large and possibly rather cumbersome DFA.

With this paper, we hope to have been able to point out interesting connections between issues in dialogue modelling on the one hand, and well-known machine models from the theory of computation on the other. There is a variety of possible avenues of future research in this area, maybe the most obvious being to try to identify further dialogue phenomena that require protocol models not covered by our work so far. Another interesting issue is the interaction between protocols and other aspects of dialogue modelling. For instance, the choice of a particular protocol model may restrict the possible dialogue strategies. Vice versa, the need for a particular strategy may influence the data type used to model the protocol. This is an exciting area of research that requires a strong interdisciplinary approach. The relevant disciplines include applied logic (specification theory), linguistics (dialogue modelling), theoretical computer science (automata theory), and artificial intelligence (multiagent systems).

Acknowledgements

A preliminary version of this paper has been presented at the 5th International Tbilisi Symposium on Language, Logic and Computation (Fernández and Endriss, 2003). We are grateful to the audience of this meeting and to an anonymous JoLLI reviewer for their feedback. Thanks are also due to Jonathan Ginzburg and Nicolas Maudet for useful comments and encouragement.

References

Aho, A. V., J. E. Hopcroft, and J. D. Ullman: 1983, *Data Structures and Algorithms*. Addison-Wesley.

Amgoud, L., N. Maudet, and S. Parsons: 2000, 'Modelling Dialogues using Argumentation'. In: *Proceedings of 4th International Conference on MultiAgent Systems (ICMAS-2000)*. IEEE Press.

Asher, N.: 1993, *Reference to Abstract Objects in English: A Philosophical Semantics for Natural Language Metaphysics*, Studies in Linguistics and Philosophy. Kluwer Academic Publishers.

Asher, N.: 1998, 'Varieties of Discourse Structure in Dialogue'. In: *Proceedings of the 2nd Workshop on the Semantics and Pragmatics of Dialogue (Twendial'98)*. Twente, The Netherlands.

Asher, N. and A. Lascarides: 1998, 'Questions in Dialogue'. *Linguistics and Philosophy* **21**(3), 237–309.

Bohlin, P., J. Bos, S. Larsson, I. Lewin, C. Matheson, and D. Milward: 1999, 'Survey of Existing Interactive Systems'. Deliverable D1.3, The TRINDI Project.

Bos, J., E. Klein, O. Lemon, and T. Oka: 2003, 'DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture'. In: *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*. Sapporo, Japan.

Carlson, L.: 1983, *Dialogue Games*, Synthese Language Library. D. Reidel.

Clark, H. H.: 1996, *Using Language*. Cambridge University Press.

Cohen, P. and H. Levesque: 1990, 'Rational Interaction as the Basis for Communication'. In: P. Cohen, J. Morgana, and M. Pollack (eds.): *Intentions in Communication*. MIT Press.

Colombetti, M.: 2000, 'A Commitment-based Approach to Agent Speech Acts and Conversations'. In: *Proceedings of the Agents-2000 Workshop on Agent Communication (AC-2000)*. Barcelona, Spain.

Endriss, U.: 2005, 'Temporal Logics for Representing Agent Communication Protocols'. In: *Proceedings of the AAMAS Workshop on Agent Communication (AC-2005)*. Utrecht, The Netherlands.

Endriss, U., N. Maudet, F. Sadri, and F. Toni: 2003, 'Protocol Conformance for Logic-based Agents'. In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*. Morgan Kaufmann Publishers.

Fernández, R.: 2003, 'A Dynamic Logic Formalisation of Inquiry-Oriented Dialogues'. In: *Proceedings of the 6th CLUK Colloquium*. Edinburgh, Scotland.

Fernández, R. and U. Endriss: 2003, 'Towards a Hierarchy of Abstract Models for Dialogue Protocols'. In: *Proceedings of the 5th International Tbilisi Symposium on Language, Logic and Computation*. Tbilisi, Georgia.

FIPA: 2002, 'Communicative Act Library Specification'. Foundation for Intelligent Physical Agents (FIPA). `http://www.fipa.org/specs/fipa00037/`.

Ginzburg, J.: 1996, 'Interrogatives: Questions, Facts, and Dialogue'. In: S. Lappin (ed.): *Handbook of Contemporary Semantic Theory*. Blackwell Publishers.

Ginzburg, J.: 2006, 'A Semantics for Interaction in Dialogue'. Manuscript in preparation for CSLI Publications and University of Chicago Press. Draft chapters are available at `http://www.dcs.kcl.ac.uk/staff/ginzburg/`.

Ginzburg, J. and R. Fernández: 2005a, 'Action at a Distance: The Difference between Dialogue and Multilogue'. In: *Proceedings of the 9th Workshop on the Semantics and Pragmatics of Dialogue (Dialor'05)*. Nancy, France.

Ginzburg, J. and R. Fernández: 2005b, 'Scaling up from Dialogue to Multilogue: Some Principles and Benchmarks'. In: *Proceedings of the 43rd Meeting of the Association for Computational Linguistics (ACL-2005)*. Ann Arbor, Michigan.

Goldblatt, R.: 1992, *Logics of Time and Computation*. CSLI Publications, 2nd edition.

Grosz, B. and C. Sidner: 1990, 'Plans for Discourse'. In: P. Cohen, J. Morgana, and M. Pollack (eds.): *Intentions in Communication*. MIT Press.

Hamblin, C. L.: 1970, *Fallacies*. London: Methuen.

Hopcroft, J. E., R. Motwani, and J. D. Ullman: 2001, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2nd edition.

Jones, A. J. I. and X. Parent: 2004, 'Conventional Signalling Acts and Conversation'. In: *Advances in Agent Communication*. Springer-Verlag.

Kreutel, J. and C. Matheson: 1999, 'Modelling Questions and Assertions in Dialogue Using Obligations'. In: *Proceedings of the 3rd Workshop on the Semantics and Pragmatics of Dialogue (Amstelog'99)*. Amsterdam, The Netherlands.

Larsson, S.: 2002, 'Issue-based Dialogue Management'. Ph.D. thesis, Göteborg University, Department of Linguistics.

Larsson, S., P. Ljunglöf, R. Cooper, E. Engdahl, and S. Ericsson: 2000, 'GoDiS: An Accommodating Dialogue System'. In: *Proceedings of ANLP/NAACL-2000 Workshop on Conversational Systems*. Seattle, Washington.

Larsson, S. and D. Traum: 2000, 'Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit'. *Natural Language Engineering* **6**(3–4), 323–340. Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering.

Levinson, S. C.: 1983, *Pragmatics*. Cambridge University Press.

Lewin, I.: 1998, 'The Autoroute Dialogue'. Technical Report CRC-073, SRI International, Cambridge Computer Science Research Centre.

Lewis, D.: 1979, 'Score-keeping in a Language Game'. *Journal of Philosophical Logic* **8**, 339–359.

Lewis, H. R. and C. H. Papadimitriou: 1998, *Elements of the Theory of Computation*. Prentice-Hall International, 2nd edition.

Mann, W. and S. Thompson: 1987, 'Rethorical Structure Theory: A Framework for the Analysis of Texts'. Technical Report RS-87-185, University of Southern California, Marina del Rey, Information Sciences Institute.

Matheson, C., M. Poesio, and D. Traum: 2000, 'Modelling Grounding and Discourse Obligations Using Update Rules'. In: *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*. Seattle, Washington.

Parsons, S., N. Jennings, and C. Sierra: 1998, 'Agents that Reason and Negotiate by Arguing'. *Journal of Logic and Computation* **8**(3), 261–292.

Pitt, J. and A. Mamdani: 1999a, 'A Protocol-based Semantics for an Agent Communication Language'. In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-1999)*. Morgan Kaufmann Publishers.

Pitt, J. and A. Mamdani: 1999b, 'Communication Protocols in Multi-Agent Systems'. In: *Proceedings of the Agents-1999 Workshop on Specifying and Implementing Conversation Policies*. Seattle, Washington.

Power, R.: 1979, 'The Organization of Purposeful Dialogues'. *Linguistics* **17**, 107–152.

Sadek, D.: 1991, 'Dialogue Acts are Rational Plans'. In: *Proceedings of the ESCA/ETR Workshop on Multi-modal Dialogue*. Maratea, Italy.

Schegloff, E. A. and H. Sacks: 1973, 'Opening up Closings'. *Semiotica* **4**(7), 289–327.

Singh, M. P.: 1998, 'Agent Communication Languages: Rethinking the Principles'. *IEEE Computer* **31**(12), 40–47.

Stalnaker, R.: 1978, 'Assertion'. *Syntax and Semantics* **9**, 315–332.

Taylor, P., S. King, S. Isard, and H. Wright: 1998, 'Intonation and Dialogue Context as Constraints for Speech Recognition'. *Language and Speech* **41**(3), 493–512.

Traum, D.: 1994, 'A Computational Theory of Grounding in Natural Language Conversation'. Ph.D. thesis, University of Rochester, Department of Computer Science.

Traum, D.: 2003, 'Semantics and Pragmatics of Questions and Answers for Dialogue Agents'. In: *Proceedings of the 5th International Workshop on Computational Semantics (IWCS-2003)*. Tilburg, The Netherlands.

Traum, D. and J. Allen: 1994, 'Discourse Obligations in Dialogue Processing'. In: *Proceedings of the 32d Meeting of the Association for Computational Linguistics (ACL-1994)*. Las Cruces, New Mexico.

Traum, D., J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poesio: 1999, 'A Model of Dialogue Moves and Information State Revision'. Deliverable D2.1, The TRINDI Project.

Wright, H., M. Poesio, and S. Isard: 1999, 'Using High-level Dialogue Information for Dialogue Act Recognition using Prosody Features'. In: *Proceedings of the ESCA Workshop on Prosody and Dialogue*. Veldhoven, The Netherlands.