Abstract Models for Dialogue Protocols: A Preliminary Report

Raquel Fernández[1] and Ulle Endriss[2]

[1] Department of Computer Science, King's College London
Strand, London WC2R 2LS (UK)
Email: `raquel@dcs.kcl.ac.uk`

[2] Department of Computing, Imperial College London
180 Queen's Gate, London SW7 2AZ (UK)
Email: `ue@doc.ic.ac.uk`

Abstract

In this report, 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: Dialogue modelling, multiagent systems, communication protocols, automata theory

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.

Communication patterns are usually modelled by means of *conventional* protocols, i.e. public conventions which specify the range of possible follow-ups available to the participating agents. Although it is clear that epistemic notions such as belief and knowledge, alongside intentions, contribute to an agent's actual responses, conventional protocols have nevertheless been shown to be a powerful descriptive and explanatory means of formalising the rules of encounter

Department of Computer Science
King's College London

Technical Report TR-03-03
July 2003

that characterise coherent interaction, both in natural language dialogue as well as in dialogue between autonomous software agents.

The focus of this paper is on the formal properties of communication protocols. We examine a variety of protocols, taking inspiration from two fields: natural language dialogue modelling and multiagent systems. More specifically, we restrict ourselves to conventional protocols that characterise the set of *legal* continuations according to externally observable features, such as the agents' actual utterances. In multiagent systems, protocols designed in accordance with this criterion have recently been put forward by a number of authors (e.g. [18, 19]). This stands in marked contrast to the approach followed, for instance, by FIPA [8] where legality conditions are explained in terms of the mental attitudes of the agents participating in a dialogue.

As we shall see, in dialogue interaction one can identify several features that increase the complexity of the dialogue structure in different ways. In our approach, this variety of phenomena motivates a hierarchy of abstract models for protocols. 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. For the dialogues that we consider in this paper, utterances are assumed to occur sequentially. This is a common assumption made in natural language dialogue modelling, whereas multiagent systems research has also tried to address concurrent communication. Also, our dialogues will typically only involve two participants.

As a starting point, in Section 2, 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. In particular, in Section 3, we augment the basic model by a *stack* component to be able to represent protocols that can handle embedded dialogues. This kind of enrichment is generalised in Section 4 to a class of *protocols with memory*. The subsequent sections discuss further instances of this class of models. In Section 5 we extend the model of Section 3 by introducing protocols with a *stack of sets* to also account for compound moves within a single turn. Section 6 discusses protocols with a simple set, which allow us to represent the kind of *blackboard architecture* used in argumentation systems. The final example for our protocols with memory are the protocols equipped with a *list* presented in Section 7. A restriction of the basic automata-based model which allows for a simple logic-based representation of protocols is given in Section 8. Finally, our conclusions are presented in Section 9.

2 Protocols as finite automata

Deterministic finite automata (DFAs) have been widely used to represent communication protocols, in particular in the area of multiagent systems. Pitt and Mamdani [18] give several examples for such 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. The following diagram provides an intuitive description of this protocol:

$$\begin{tikzcd}
{} \arrow[r, Rightarrow] & 0 \arrow[r, "A\colon \mathit{inform}(c)"] & 1 \arrow[r, bend left, "B\colon \mathit{acknowledge}"] \arrow[d, "B\colon \mathit{end}"'] & 2 \arrow[l, bend left, "A\colon \mathit{inform}(c)"] \arrow[d, "A\colon \mathit{end}"] \\
 & & 3 & 4
\end{tikzcd}$$

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 *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.

This type of protocols 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 (see e.g. [13, 16]) using a terminology appropriate for the description of dialogue protocols. 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* $\mathcal{L}$,[1] and a *transition function* $\delta : Q \times \mathcal{L} \to Q$. The elements of $\mathcal{L}$ are called *utterances* and are constructed from a finite set $\mathcal{A}$ of *agents* (or *dialogue participants*), a finite set $\mathcal{M}$ of *dialogue moves* (or *communicative 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

[1]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. We remark here that the input alphabet is usually taken to be finite [13], while our representation does not exclude infinite alphabets (although any given DFA will necessarily only make use of a finite subset of the alphabet).

are going to consider typically only involve two participants we may think of $\mathcal{A}$ as the set $\{A, B\}$.

This representation allows for a simple formalisation of the notion of *legality* of an utterance at a given point in a dialogue. Given the current dialogue state q, an utterance u constitutes a legal continuation 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. A legal dialogue is a dialogue where each utterance is a legal continuation of the preceding sequence of utterances. Furthermore, a legal dialogue is complete iff it results in one of the final states in F.

3 Protocols that allow for subdialogues

DFA-based protocols have also been successfully used in natural language interaction, usually under the name of *conversational games* (e.g. [15]). However, some very common features of natural language dialogue cannot be captured by a DFA. The following example shows a dialogue where several question-answer sequences are embedded:

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

Replying to a question with another question (2) and asking for clarification (3) are rather common phenomena in natural language dialogue, especially in information-oriented interaction. Thus, the presence of embedded subdialogues 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. The machine model of a DFA together with a stack corresponds to a *pushdown automaton*, at least if we restrict ourselves to a finite communication language [13].

An example of a structuring mechanism able to handle this kind of phenomena is Ginzburg's QUD (*questions under discussion*) [9, 10]. Simplifying a little, in Ginzburg's approach questions get introduced into QUD and get *downdated* once they are answered. 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.

Another approach that provides a similar structuring mechanism is the modelling of dialogue by means of *discourse obligations* [20, 14]. In this case, the authors suggest that a question imposes an obligation on the addressee to answer the question. Once the question is answered, the obligation is discarded.

4 Protocols with memory

Both discourse obligations and questions under discussion are examples for *abstractions* from the full dialogue history. We only keep those parts of the history that are relevant to the legality of future utterances, in a convenient format. For the examples of the previous section, this format has been that of a stack.

DFAs are abstract machines with a limited amount of memory (encoded by the states of the automaton). Adding a (finite) stack as discussed earlier 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, sets or lists [1]. 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 recent formalisation of a complex protocol with memory in dynamic logic (DL), namely a protocol for inquiry-oriented dialogues based on Ginzburg's *dialogue gameboard* [7], suggests that the usual DL program constructors provide an adequate language to combine these basic operations.[2]

In the following sections, we are going to discuss several examples that motivate different choices for ADTs as memory components on top of our basic DFA-based model.

5 Protocols with a stack of sets

In Section 3, 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. As some authors have noticed [10, 4], however, when successive queries are asked by a single speaker, this simple kind of protocol is not always correct. This is illustrated by the following example (adapted from [4]):

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

Dialogues 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

[2]In dynamic logic, complex programs can be constructed from basic ones (such as e.g. *pop*) via the operations of *composition* (;), *choice* ($\cup$), and *iteration* (*) [11]. Additionally, the DL programming language includes the *test* operator ? which may, for instance, be applied to expressions over functions such as *top* provided by the ADT in question.

them in any order. In such dialogues, the questions under discussion are in what has been called a *coordinate structure* [4], 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.

In fact, although in Section 3 we have represented 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.

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 on 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.

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 example above), while the second one is executed when a different speaker replies to a question with another question (as in the example in Section 3). The following dialogue (taken from [10]), however, shows that successive querying within a single turn does not always involve question coordination:

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

Notice that here the first question asked takes precedence over the last one—only after the first question is answered does the second question get addressed. Indeed, answering (2) before (1) would seem rather strange in this situation.

Ginzburg uses examples like the one above 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 our abstract model 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.

Which mechanism is used to add a question to QUD does then not only depend on whether successive questions are asked within a single turn. As Ginzburg [10] and Asher [4] argue, following the *Discourse Relations* approach,[3] when two questions are asked by the same speaker, how the second question gets added to QUD depends on its relation to the first one. Under this view, in the first example of the this section, the discourse relation of *Coordination* is said to hold between questions (1) and (2), while in the second example the relation that holds between the questions is that of *Query-extension*. When *Coordination* holds, the respondent can choose to answer the questions in any order. If, on the oder hand, the questions are related by *Query-extension*, it is more appropriate to answer the first question posed first.

When the last question asked takes precedence (as in the example in Section 3), the relation that holds between the questions is that of *Query-elaboration*. This can also happen when the queries are asked by a single speaker, as the following dialogue (taken from [10]) shows:

A :	(1) Who have you invited?	$[Q_1]$
A :	(2) Have you invited Jill?	$[Q_2]$
B :	(3) Yes.	$[A_2]$
A :	(4) Aha.	$[Ack]$
B :	(5) I'm also inviting Merle and Tawfik.	$[A_1]$

All these examples show that, in order to determine the legality of a dialogue move with respect to a given protocol, one also has to take into account complex relations between the elements of the content language $\mathcal{C}$. Integrating this kind of conditions with our abstract model of protocols with memory is one of issues we are currently investigating further.

6 Protocols with a blackboard

Our next example is inspired by work on argumentation in discourse modelling. Argumentation-based protocols have recently been used to model different types of dialogues (e.g. negotiation dialogues) between software agents [2]. Central to this approach is the notion of a so-called *commitment store* [12]. For example, *asserting* an argument amounts to placing that argument into one's commitment store. A *retract* move would then be considered legal only if the corresponding argument can be found in the agent's commitment store (and would itself

[3]During the last decade, inspired by the early work of Mann and Thompson [17], a lot of attention has been paid to characterising the nature of *Discourse Relations*, i.e. coherence relations that can hold between adjacent sentences, mostly in text/monologue (e.g. [3]), but recently also in dialogue [5].

cause the respective argument to be deleted again). 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.

It is interesting to note that adding a set to a DFA does not increase expressive power provided we are working with a *finite* memory alphabet, because the range of all possible configurations of the set component could be encoded into a larger DFA.[4] However, such a construction would result in an exponential blow-up of the set of states; that is, a blackboard architecture can have very practical 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.

We also note here that, in contrast to the case of sets, enriching a DFA-based protocol with a stack *does* result in an increase in expressive power, even if the memory alphabet is required to be finite. This immediately follows from the fact that pushdown automata are strictly more powerful than DFAs [16]. Furthermore, adding a second 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 [16].

7 Protocols with a list

As a final example for a protocol with memory, we remark that 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 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).

This is the most powerful protocol model we have discussed, because, given the (full) dialogue history, it should—in principle—always be 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

[4]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.

(in multiagent systems research): what is legal may only depend on publicly observable facts. 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) does not seem particularly supportive for designing concise protocols.

A DFA together with a list can be used to simulate a Turing machine. To see this, recall that for any Turing machine we can construct a pushdown automaton with two stacks that accepts the same language [16]. We can use the list to encode two stacks as follows. At the beginning, the list is initialised with a single special symbol that is different from any of the symbols in the memory alphabet used for the rest of the simulation. Then, to push an element onto the first stack we append that element to the beginning of the list; to push an element onto the second stack we append it at the end of the list. Similarly, popping the first stack means removing the first element of the list; popping the second stack amounts to removing the final element of the list. Whenever the first or the last element in the list is the special symbol used to initialise the list, then the respective popping operation will fail (because the respective stack would be empty).

8 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 [6]. 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. 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):[5]

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

While such a simplistic approach to representing protocols will have little relevance to natural language dialogue modelling, it can be of great interest in the area of multiagent systems. As shown in [6], 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 (at least in the sense of the agent being guaranteed not to utter anything illegal; guaranteeing that an agent actually utters anything at all appears to be more difficult).

[5]The *next-operator* $\bigcirc$ (familiar from linear temporal logic [11]) refers to the next turn in the dialogue.

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). An example for a non-shallow protocol would be a DFA where two edges with the same label point to two different states. In principle, it is always possible to turn a DFA-based protocol into a shallow protocol by renaming any duplicate transitions.[6] In a context where renaming is not acceptable, however, shallow protocols really do determine a proper subclass to DFA-based protocols.

9 Conclusion

In this report, we have reviewed 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. 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.

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 model actual human-human dialogues. For instance, pushdown automata, which we have used to model the phenomenon of subdialogues in Section 3, are only strictly more expressive than simple DFAs if the size of the stack is not bounded (provided the memory alphabet is finite). 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 not more expressive than a simple DFA (again, provided the memory alphabet is finite). 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, then a single large and possibly rather cumbersome DFA.

[6]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. In such cases, it seems advantageous to use the simpler model of shallow protocols in the first place.

Acknowledgements. The work of the first and the second author is supported by UK ESRC grant RES-000-23-0065 and EC grant IST-2001-32530, respectively. Thanks to Jonathan Ginzburg and Nicolas Maudet for helpful comments and encouragement.

References

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

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

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

[4] N. Asher. Varieties of Discourse Structure in Dialogue. In *Proceedings of the Second International Workshop on Dialogue (Twendial)*, 1998.

[5] N. Asher and A. Lascarides. Questions in Dialogue. *Linguistics and Philosophy*, 21:237–309, 1998.

[6] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Protocol Conformance for Logic-based Agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, 2003. To appear.

[7] R. Fernández. A Dynamic Logic Formalisation of Inquiry-Oriented Dialogues. In *Proceedings of the 6th CLUK Colloquium*, Edinburgh, 2003.

[8] Foundation for Intelligent Physical Agents (FIPA). *Communicative Act Library Specification*, 2002. http://www.fipa.org/specs/fipa00037/.

[9] J. Ginzburg. Interrogatives: Questions, Facts, and Dialogue. In S. Lappin, editor, *Handbook of Contemporary Semantic Theory.* Blackwell Publishers, Oxford, 1996.

[10] J. Ginzburg. A Semantics for Interaction in Dialogue. Forthcoming for CSLI Publications. Draft chapters are available from http://www.dcs.kcl.ac.uk/staff/ginzburg/.

[11] R. Goldblatt. *Logics of Time and Computation.* CSLI, 2nd edition, 1992.

[12] C. L. Hamblin. *Fallacies.* Methuen, London, 1970.

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

[14] J. Kreutel and C. Matheson. Modelling Questions and Assertions in Dialogue Using Obligations. In *Proceedings of Amstelog'99, the 3rd Workshop on the Sematics and Pragmatics of Dialogue*, Amsterdam, 1999.

[15] I. Lewin. The Autoroute Dialogue. TRINDI Deliverable, SRI International, 1998.

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

[17] W. Mann and S. Thompson. Rethorical Structure Theory: A Framework for the Analysis of Texts. Technical Report RS-87-185, Information Systems Institute, 1987.

[18] J. Pitt and A. Mamdani. 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, 1999.

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

[20] D. Traum and J. Allen. Discourse Obligations in Dialogue Processing. In *Proceedings of the 32nd Annual Meeting of the ACL*, 1994.