Towards a Hierarchy of Abstract Models for Dialogue Protocols

Raquel Fernández[*] **Ulle Endriss**[†]

Abstract

In this paper, 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.

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 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. [14,

* Department of Computer Science, King's College London, `raquel@dcs.kcl.ac.uk`
† Department of Computing, Imperial College London, `ue@doc.ic.ac.uk`

15]). This stands in marked contrast to the approach followed, for instance, by FIPA [6] 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 [14] 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:

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

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' such that the automaton's transition function maps the pair (q, u) to q'. This type of protocols will provide the starting point for our proposed classification of communication protocols.

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. [12]). 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.	$[A_1]$

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* [13].

An example of a structuring mechanism able to handle this kind of phenomena is Ginzburg's QUD (*questions under discussion*) [7, 8]. 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* [16, 11]. 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*. 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* [5], suggests that the usual DL program constructors provide an adequate language to combine these basic operations.[1]

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

As some authors have noticed [8, 3], when successive queries are asked within a single turn, a protocol with a simple stack is not always correct. This is illustrated by the following example (adapted from [3]):

1. 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* (*) [9]. 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.

A :	(1)	Where were you on the 15th?	[Q_1]
A :	(2)	Did you talk to him 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 them in any 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.

An interesting issue is what causes a question to be either inserted into the maximal set of the stack or, alternatively, to be pushed on top of the stack of sets. As Ginzburg [8] and Asher [3] argue, this depends on the relation of that question to those that are already in the maximal set on the stack. For instance, in the previous example the discourse relation of *Coordination* is said to hold between questions (1) and (2), while in the earlier example the relation that holds between the questions is that of *Query-elaboration*. This shows 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 utterances occurring in a dialogue. Integrating this kind of conditions with our abstract model of protocols with memory is an issues that requires further investigation.

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 between software agents [2]. Central to this approach is the notion of a so-called *commitment store* [10]. 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 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.

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). Note that a DFA together with a list can be used to simulate a Turing machine (the list may be thought of as the machine's tape).

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 (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.

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 [4]. 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:[2]

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

While such a simplistic approach will have little relevance to natural language dialogue modelling, it can be of great interest in the area of multiagent systems.

2. The *next-operator* $\bigcirc$ (familiar from linear temporal logic [9]) refers to the next turn in the dialogue.

As shown in [4], 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.

9 Conclusion

In this paper, 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.

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 useful comments and encouragement, and to the referees of the Tbilisi Symposium 2003 for their helpful suggestions.

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. Varieties of Discourse Structure in Dialogue. In *Proceedings of the 2nd Workshop on the Semantics and Pragmatics of Dialogue (Twendial)*, 1998.

[4] 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)*. Morgan Kaufmann Publishers, 2003.

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

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

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

[8] 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/.

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

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

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

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

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

[14] 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.

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

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