Using Generalised Dialogue Models to Constrain Information State Based Dialogue Systems

Robert J. Ross John Bateman* Hui Shi
Collaborative Research Centre SFB/TR 8 Spatial Cognition
Universität Bremen
Bibliothekstr. 1, 28334 Bremen, Germany
{robertr,shi}@informatik.uni-bremen.de
*bateman@uni-bremen.de

Abstract

While Information State (IS) based techniques show promise in the construction of flexible, knowledge-based dialogue systems, the many declarative rules that are used to encode Dialogue Theories often lead to opaque systems that are difficult to test and potentially unintuitive to users. In this paper, we advocate the application of explicitly defined Generic Dialogue Models (GDMs), encoded as recursive transition networks (RTNs), to the structuring of information state-based dialogue managers. To this end, we review the state of GDM approaches, comparing and contrasting them against the Dialogue Theories which are typically implemented using information state approaches. Furthermore, to support our approach, we present an extension of the ALPHA (A Language for Programming Hybrid Agents) language, which has been enhanced to support information state and GDM concepts directly.

1 Introduction

The Information State (IS) based approach to dialogue management [14, 28] advocates dialogue manager construction based around discourse objects (e.g., questions, beliefs) and rules which encode update-relationships between these objects. As such, IS based systems may be viewed as practical instantiations of agent-based models, instantiations where the broad notions of beliefs, actions, and plans, are replaced with more precise semantic types and their inter-relationships. More specifically, the IS approach provides a programming paradigm for dialogue managers which centres around a well-structured information state model – a model that works with classes of declarative rules which effect information state transitions. Indeed, as a modelling or implementation paradigm, the IS approach does not impose any one discourse model or theory, but instead allows for the development of dialogue managers based upon any number of dialogue theories ranging from finite state to Larsson's Issue Based Models [13].

While the IS approach's flexibility allows for the capturing of a wide range of Dialogue Theories, this same flexibility can also give rise to the construction of opaque and unintuitive dialogue managers. Following initial studies into the use of IS based dialogue managers in safe-system construction [21], four deficiencies of IS implementations have become apparent:

- Opacity of Control – As with all declarative rule based implementations, the use of a potentially large number of rules to define information state transitions can lead to systems that are difficult to design and debug, with unforeseen logic errors which are tedious to trace, and which may lead to potentially serious side-effects.
- Low Empirical Justification – Furthermore, while the use of information state and transition rules provide a quick means for managing dialogue state, the dialogue models implicitly built into IS implementations may become unintuitive to users, producing unnatural dialogic interactions, and degrading the perceived system performance.

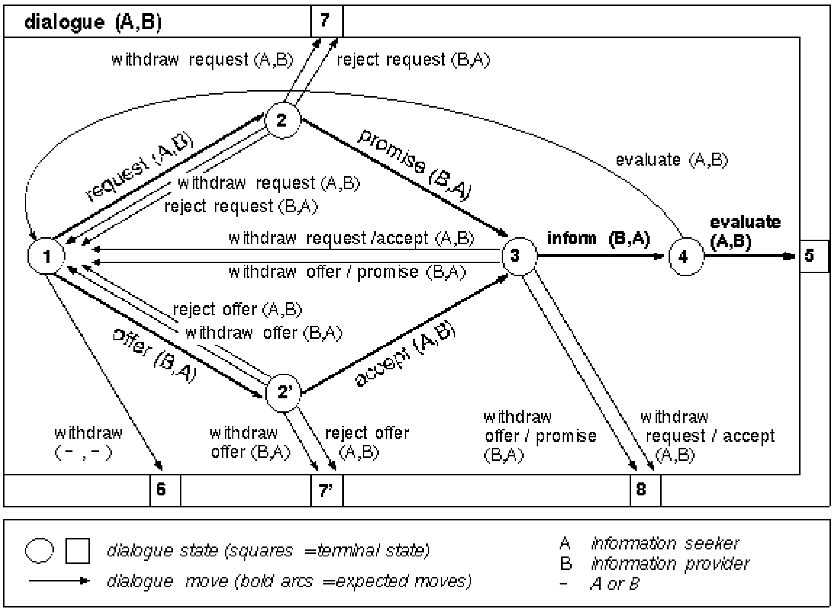

Figure 1: The Conversational Roles Model. Taken from [25]

- No Look-ahead – As claimed by Traum & Larsson, the generation of expectation values for the language understanding and contextualization processes is an important role for dialogue management [28]. Unfortunately, the use of simple data types and declarative rules makes the handling of expectations directly within IS implementations difficult.
- Limited Tool Support – Finally, on a practical note, IS based toolkits still provide a limited functionality, particularly with regard to rapid prototyping, code re-use, and debugging.

In the following sections, we motivate an approach to the integration of formally specified Generic Dialogue Models (GDMs) into Information State implementations – thus addressing the first three points of opacity of control, user satisfaction, and expectation generation. The result is a dialogue management model that retains the level of expressiveness provided by the current IS approach, while benefiting from the structuring guidance of generalised dialogue models. This approach is motivated by the observation that IS implementations can often include a large set of rules, of which only a few may directly relate to the encoding of the underlying dialogue model, while others relate to the broader discourse theory under investigation. Our approach allows us to extract and analyse the dialogue models underlying these dialogue theory implementations, thus allwing the study of the user observable properties of that dialogue theory.

We begin in Section 2 with an introduction to GDMs, focusing on pattern based approaches, and contrasting these relatively simple models with the more sophisticated Dialogue Theories that are traditionally encoded in information state implementations. Then, in Section 3, we further examine the relationship between the Information State paradigm and Generalised Dialogue Models, and propose a methodology for the application of GDMs to IS implementations. To support this methodology, Section 4 then introduces ALPHA, an extended agent-oriented programming language that has been developed to handle aspects of information state and GDMs alongside the more powerful BDI (Belief Desire Intention) notions of explicit commitment, goal, and plan modelling within a complete cognitive agent. Finally, in Section 5, we briefly review related work before summarising and looking towards future activities.

2 Generalised Dialogue Models

While some recent work has aimed to simplify IS based toolkits, e.g. DIPPER [1], resultant systems are still based on large sets of declarative rules that can mask underlying dialogue models. This problem, compounded by a frequent gulf between discourse modelling and dialogue management [30], means that the dialogue models underlying an IS implementation are often overlooked as both a structuring property for the implementation, and a means of assessing the quality of an implemented system from the human-computer interaction perspective.

A tighter coupling between intended dialogue model and information state implementation provides a structuring mechanism for dialogue manager implementation and analysis. Generalised Dialogue Models (GDMs) offer a situation and dialogue theoretic independent manner of providing this tighter coupling. Informally, we define GDMs as theories of dialogue structure that are independent of domain application and which explictly define the underlying dialogue grammars without reference to either domain specifics or internal mental state dynamics. On the other hand, Dialogue Theories, e.g., Ginzburg's Dialogue Game Board [8], capture complete models of both interaction and the internal dialogue modelling. Naturally such Dialogue Theories often encompass an underlying dialogue model, often 'generalised', but this is more often implicit and difficult to distinguish from the greater Dialogue Theory. Thus, GDMs are closer in nature to the finer-grained dialogue grammars often captured in Conversational Game Theory [17].

We take the view of [30], that GDMs can be broadly categorised into two groups: pattern based models and plan based models. In pattern-based models, recurrent interaction patterns or regularities in dialogue at the illocutionary force level of speech acts are identified [25]. While, in the second approach, i.e. plan-based models, dialogue is modelled in terms of speech acts and their relation to plans and mental states in the greater agent design [2]. Pattern based models describe what happens, but care little about why. Conversely, plan-based models contextualize speech acts within the greater agent plans and rationality, but are costly and consider the actual patterns as more epiphenomenal on the 'real' reasons for the dialogic interaction, i.e., the underlying plans.

This separation corresponds broadly to different theoretical positions taken with respect to discourse and dialogue as such. The plan-based model, prominent in cognitive approaches, emphasises the 'behind-the-scenes' planning in terms of beliefs and goals; the dialogue transition model aligns more naturally with interaction-based approaches that emphasise the ongoing achievement of dialogue in the produced utterances of the speakers. This latter approach is dominant in, for example, conversation analysis and approaches descended from it.

As with models used in Conversational Game Theory, individual GDMs are often formalised in terms of a number of transition networks, where any dialogue move, or transition in the network, can be unfolded to involve a transition network of its own: the network is thus a recursive transition network (RTN). To illustrate the characteristic properties of a GDM, the following briefly reviews one prominent GDM, the COR Model.

2.1 Sitter & Stein's COR Model

Of the pattern based GDMs, one which has already been used extensively in the construction of 'generation centric' dialogue systems (e.g., MERIT [6], MIRACLE [26] and SPEAK! [27]) is the '**Co**nversational **R**oles' COR Model of Sitter & Stein [25]. Like its forerunner, Winograd & Flores' '**C**onversation **f**or **A**ction (CfA) model [29], the COR model was originally set out as a communicative-based approach to interaction in the relatively limited context of information-seeking dialogues. Despite this, critical features of dialogues such as the ability of speakers to interrupt, to answer questions with questions, to change topics, to jump ahead to provide answers to questions that had not already been asked, were already included in the basic structure of the GDM.

The COR model, partially depicted in Figure 1, has been defined as a RTN consisting of a main dialogue along with a number of sub-dialogue structures, each of which represents individual dialogue moves within the complete RTN. An individual dialogue move can therefore involve almost arbitrarily complex sub-dialogues of its own: this is seen at many points in the definition of the model where the basic dialogue network is re-used by particular steps in sub-dialogues. The most comprehensive COR network, and that which interlocutors are always making at least one transversal of, is the basic dialogue. The transitions between states in this network are themselves transition networks, and consist of sub-dialogue networks including Offer, Request and Accept. Moreover, where each dialogue network consists of a potentially complex dialogue move, these networks can also consist of arcs corresponding to simple dialogue moves akin to dialogue acts.

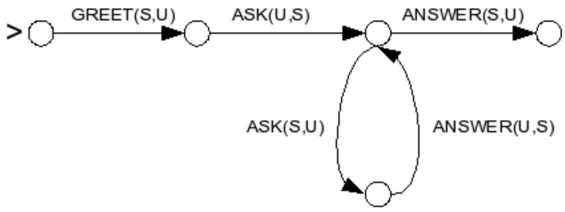

Figure 2: Abstraction of IBiS1 Dialogue Model

3 Generalised Dialogue Models & Information State

In the last section, we reviewed the basic nature of GDMs, setting them apart from the more complex Dialogue Theories which are often implemented with an IS approach. Here, we examine the relationship between IS based implementations and GDMs, proposing a general methodology for the use of GDMs with IS implementations. To establish this relationship, we must first make clear the distinction between the information state based dialogue management model or paradigm, and the dialogue theories which can be implemented with such a paradigm. Broadly, we share the view that as a paradigm, the Information State based approach is extremely flexible and can support the implementation of a wide variety of dialogue theories. Such implementations range from simple finite state models using registers and state transition rules, to what we refer to as *IS centric dialogue theories* where the dialogue modelling approach is inextricably linked to the modelling of the dialogue's information state. Examples of such theories include those models behind GoDiS, and EDIS [28].

In investigating the relationship between the IS paradigm and GDMs encoded as Recursive Transition Networks (RTNs), it is important to distinguish between the encoding of RTNs through information state, and the use of RTNs in information state. The former of these two approaches refers to the fact, as observed in [28], that recursive transition networks can be directly encoded through an information state based implementation through the use of a stack to record a history of nested state positions, and a collection of update rules to encode state transitions. The latter view, however, reflects the use of RTNs as part of the data types used to store information state; this being analogous to the use of queues, records or predicate sets. It is the latter view of using RTNs within the information state that can best leverage off existing generalised dialogue models.

By directly embedding GDMs within the information state, various limitations of current IS implementations, including expectation provision and empirical quality, can be overcome. Specifically, a clear model of expected discourse moves can be extracted from the recursive transition network that encodes a GDM. Thus, applying a similar approach to Lemon's use of allowed attachments [16], the search space for intention identification can be considerably reduced. Also, abstraction of the many declarative rules that constitute an information state implementation can make evaluation of the *quality* of the underlying dialogue model more straightforward. Furthermore, through simulation, the accuracy of rules in an information state based implementation can be judged against the sought-after generalised dialogue model. Moreover, when considered in the light of the ever increasing application of SDS to *safety critical* applications such as service robotics and automotives [12, 11], the need for analysis and verification of dialogue models underlying spoken dialogue systems becomes even more imperative.

GDMs can either be applied to the formulation of information state type dialogue models, or, abstracted from an otherwise implemented dialogue theory. For example, despite the apparent complexity of the dialogue theories behind the IBIS series of dialogue managers [13], an underlying GDM structure is easily extracted from the complete dialogue theory. This can be done by examining selection and update rules with regard to the movement of information on and off the latest utterance record of the IBIS's information state, i.e., `/SHARED/LU/MOVE`. For example, Figure 2 depicts an abstraction of IBiS1's underlying dialogue model, which, once extracted, can be used for explicit analysis against empirically derived dialogue models, or in the production of expectation values as described in the following section.

4 Applying Generalised Dialogue Models to ALPHA

To demonstrate the applicability of the approach outlined in the last section, we now outline an extension of the ALPHA (A Language for Programming Hybrid Agents) programming language which directly combines the knowledge and dialogue structuring properties of information state and GDMs respectively, with a fully fledged BDI (Belief, Desire, Intention) style architecture.

ALPHA is a modern Agent Oriented Programming language with deliberative, imperative, and role re-use features. ALPHA is a fully supported language with a run-time environment and interpreter provided by the open-source *Agent Factory* Framework [3, 4] [1]. While ALPHA has been historically developed as a generic agent-oriented programming language in the style of Agent-0 [24] and 3APL [5], recent extensions of the language have focused on developing a theory of knowledge rich cognitive discourse agents. Here we will limit discussion to a general overview of the language and a description of the features directly related to information state and GDM representation. For a detailed description of the language's syntax, semantics, and use, the user is directed elsewhere [20, 19].

4.1 ALPHA : Basic Agent Structure & Execution

From the object perspective, an ALPHA agent is defined as a tuple of mental objects:

$$Agent = \{\mathcal{IS}, \mathcal{G}, \mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{IR}, \mathcal{RR}, \mathcal{CR}, \mathcal{PR}\}$$

where $\mathcal{IS}$ is the agent's Information State, i.e, the knowledge held by the agent about itself or its environment, which can range in complexity from a single atom or set of temporal beliefs to a highly structured knowledge representation; $\mathcal{G}$ is the agent's Goal Set, each of which are states of the world that the agent wishes to bring about; $\mathcal{C}$ is the agent's Commitment Set, each of which represent a promise made by the agent to itself or others; $\mathcal{A}$ is the agent's Actuator Set, comprising the most basic actions that can be performed by an agent (actuators are often implemented in external code such as C or Java); $\mathcal{P}$ is the agent's Perceptor Set, each of which is a block of external code that is triggered regularly to allow the agent to acquire beliefs about its environment. $\mathcal{IR}$ is the agent's Inference Rule Set, which enable the agent to infer new knowledge from its information state. $\mathcal{RR}$ is the agent's Reactive Rule Set, which triggers an action or plan in the event that some belief is entailed by the agent's information state. $\mathcal{PR}$ is the agent's set of Plans constructed from other plans or actuators; Finally, $\mathcal{CR}$ is the agent's Commitment Rule Set, each of which allow an agent to adopt a commitment to action if some condition is entailed by $\mathcal{IS}$. ALPHA does not contain primitives for communication. Instead, appropriate actuators and perceptors can be built to implement any communication strategy. For example, runtime libraries exist to support standard-compliant agent communication such as the FIPA Agent Communication Language (ACL) [7].

While plans give ALPHA some superficial features of an imperative programming language, i.e., an ALPHA agent can execute a structured sequence of actions, ALPHA is, at the highest level, a declarative programming language. An ALPHA agent's Life Cycle is built out of a number of Execution Cycles. During each of these execution cycles all perceptors are emptied, and reactive and commitment rules are tested, with plans or actuators 'fired' in their own threads when necessary. The use of declarative rules and simple sequencing of actions makes ALPHA superficially similar to DIPPER [1] and the TrindiKit update languages [28], but other features of the language – including role re-use, explicit commitment and goal model, and a full structuring of actuator and plans – make ALPHA a far richer language for the construction of conversational agents.

4.2 Defining and Manipulating Information State

An agent's information state denotes what the agent believes to be true about itself, other agents, or its environment. Such information may include both details of discourse related object, but also levels of general knowledge. ALPHA's information state consists of a number of containers of type temporal belief set, non-temporal belief set, atom, stack, and RTN. For example, a simple information state can be defined with the IS definition construct as follows:

[1] See http://agentfactory.sourceforge.net & http://www.agentfactory.com

```
BEGIN_IS
  BELIEF : TSET;
  PUB_BELIEF : TSET;
  LAST_MSG : ATOM;
  MSG_HISTORY : STACK;
  GDM1 : RTN;
END_IS
```

The construct defines a number of containers, which together constitute the agent's complete information state. Containers are defined in terms of a type and a name. For instance, in the example above the information state consists of five containers. The first of these containers, named `BELIEF`, is of type `TSET`, i.e., a temporal set; while the third, named `LAST_MSG`, is of type `ATOM`, and the fifth, named `GDM1`, is a recursive transition network.

In ALPHA, manipulation of the agent's mental state is facilitated via a number of actuators and 'mental state operators'. To support GDM definition and manipulation a number of new RTN operators have been introduced. Before updates or queries of GDMs can be made, RTN containers must first be defined and initialised. Once initialised, these containers can then be directly updated, or be used to produce expectation values using a number of basic operators. The provision of expectation values is a distinct advantage of explicit GDM modelling within information state; here, expectation values of a user's next utterance can substantially reduce the search space of intention recognition. The expectation values, defined as the generalised dialogue states accessable from the current dialogue state, may be made available from an `RTN` container to any part of the complete cognitive agent.

4.3 GDM Specification

For the specification of GDMs we adopt a formal methodology based on the FDMSC toolkit [23]. Using FDMSC, a formal specification of the desired dialogue model may be produced in the process specification language CSP (Communicating Sequential Processes) [10, 18]. From this formal specification, which may undergo analysis as described in [22], an XML based state machine specification is produced. This description may then be parsed by an ALPHA actuator to build up the GDM container as described in the previous section. Thus, we can avail ourselves of formal techniques in the initial specification and analysis of the dialogue model, whilst producing a more straight-forward description of the model to be held directly within the information state. Naturally, the CSP specification and analysis steps could be skipped entirely, thus, requiring that an XML specification of the state machine be constructed directly by an end developer. However, the significance of being able to formally analysis the dialogue model, particularly when considering safe-system applications [12, 11], mitigates any additional effort in the construction of the CSP specification.

5 Related Work

Another system that makes use of a form of expectation values is Lemon's Conversational Intelligence Architecture (CIA) [15, 16]. The CIA is broadly based on an Information State based approach, making use of a model of conversational state and algorithms that explicitly define the interplay between the conversational elements, including: Dialogue Move Tree, Activity Node List, Activity Tree, System Agenda and Pending List. The process of attachment in intention recognition requires that the system determines where on the Dialogue Move Tree an incoming conversational move should be placed. The attachment process makes use of a set of node classes and constraints on which conversational moves may be attached to instances of each node class. Allowed attachments implicitly define a dialogue grammar for a particular domain scenario, but unlike the work presented here, do not constitute a well-defined element within the complete information state.

6 Summary & Future Work

In this paper we have attempted to motivate the use of explicitly defined generalised dialogue models (GDMs) directly within information state based dialogue management, thus providing mechanisms for both the generation of expectations and the validation of dialogue implementations against desired dialogue models and properties. Our notion of GDMs, encoded as recursive transition networks, model the externally observable qualities of interaction in terms of dialogue acts, but abstract away from the details of internal manipulation that are often the subject of more complete Dialogue Theories. In addition to motivating the use of GDMs, we also illustrated their use through a discussion of a number of extensions that have been made to the ALPHA programming language.

The general approach presented here, gives rise to several research questions. Firstly, while our approach is not limited to any one GDM, we feel that existing models, such as the COR model, are worthy of further investigation. To this end, we wish to look at the relevance of COR, originally formulated for information-seeking dialogues, to the more general domain – particularly dialogues concerning spatial tasks in mixed-initiative systems [21]. Such investigations require empirical study of the validity of GDMs in specific application scenarios. Secondly, we believe that the applicability of GDMs extends beyond the pure information state approach to the broader class of plan-based and agent-based dialogue models. Reconciliation of the implicit dialogue models often present in task-plans – and manifested through agent intentions (c.f. [9]) – against generalised dialogue models is worthy of considerable effort.

References

[1] Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, 2003.

[2] P. R. Cohen and C. R. Perrault. Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science*, 3:177–212, 1979.

[3] Rem W. Collier. *Agent Factory: A Framework for the Engineering of Agent Oriented Applications*. PhD thesis, University College Dublin, 2001.

[4] Rem W. Collier, G.M.P. O'Hare, Terry Lowen, and C.F.B. Rooney. Beyond Prototyping in the Factory of the Agents. In *3rd Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'03)*, Prague, Czech Republic, 2003.

[5] Mehdi Dastani, Birna van Riemsdijk, Frank Dignum, and John-Jules Meyer. A Programming Language for Cognitive Agents: Goal Directed 3APL. In *Proceedings of AAMAS 03*, 2003.

[6] Stephan Dilley, John A. Bateman, Ulrich Thiel, and Anne Tissen. Integrating Natural Language Components into Graphical Discourse. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 72–79, Trento, Italy, 1992. Association for Computational Linguistics. 31 March - 3 April.

[7] Foundation for Intelligent Physical Agents. FIPA 97 specification part 2. June 24 1998.

[8] Jonathan Ginzburg. Dynamics and the Semantics of Dialogue. In J. Seligman, editor, *Language, logic and computation*, volume 1. CSLI Lecture Notes, CSLI Stanford, 1996.

[9] Barbara J. Grosz and Candace L. Sidner. Attention, Intentions and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204, July-September 1986.

[10] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[11] Bernd Krieg-Brückner, Hui Shi, and Robert Ross. A safe and robust approach to shared-control via dialogue. *Journal of Software*, 15(12):1764 –1775, 2004.

[12] Axel Lankenau and Thomas Röfer. A versatile and safe mobility assistant. *IEEE Robotics and Automation Magazine*, 7(1):29 – 37, 2001.

[13] Staffan Larsson. *Issue-Based Dialogue Management*. Ph.d. dissertation, Department of Linguistics, Göteborg University, Göteborg, 2002.

[14] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340, 2000. Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering.

[15] Oliver Lemon and Alexander Gruenstein. Multithreaded Context for Robust Conversational Interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer-Human Interaction*, 11(3):241–267, September 2004.

[16] Oliver Lemon, Alexander Gruenstein, and Peters Peters. Collaborative Activities and Multi-tasking in Dialogue Systems. *Traitement Automatique des Langues (TAL)*, 43(2):131–154, 2002. Special issue on dialogue.

[17] Ian Lewin. A Formal Model of Conversational Game Theory. In *Fourth Workshop on the Semantics & Pragmantics of Dialogue*, 2000.

[18] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1998.

[19] Robert Ross, Rem Collier, and G.M.P. O'Hare. AF-APL Bridging Princples & Practices in Agent Oriented Languages. In Afael Bordini, M. Dastani, J. Dix, and Amal El Fallah-Segrouchni, editors, *PROMAS 2004*, volume 3346 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, D-69121 Heidelberg, Germany., 2004.

[20] Robert J. Ross, Rem Collier, and G.M.P. O Hare. ALPHA A Language for Programming Hybrid Agents. Presented at Second European Workshop on Multi-Agent Systems (EUMAS 04, Dec 2004.

[21] Robert J. Ross, Hui Shi, Tillman Vierhuf, Bernd Krieg-Bruckner, and John Bateman. Towards Dialogue Based Shared Control of Navigating Robots. In *Proceedings of Spatial Cognition 04*, Germany, 2004. Springer.

[22] Hui Shi and John Bateman. Developing Human-Robot Dialogue Management Formally. In *Symposium on Dialogue Modelling and Generation*, Amsterdamn, the Netherlands., 2005.

[23] Hui Shi, Robert J. Ross, and John Bateman. Formalising control in robust spoken dialogue systems. In *Software Engineering & Formal Methods 2005*, Germany, Sept submitted.

[24] Yoav Shoham. Agent Oriented Programming. *Artificial Intelligence*, 60:51–92, 1993.

[25] Stefan Sitter and Adelheit Stein. Modeling information-seeking dialogues: The *Co*nversational *R*oles model. *Review of Information Science*, 1(1):n/a, 1996. (On-line journal; date of verification: 20.1.1998).

[26] A. Stein, J. A. Gulla, A. Müller, and U. Thiel. Conversational interaction for semantic access to multimedia information. In M. T. Maybury, editor, *Intelligent Multimedia Information Retrieval*, pages 399–421. AAAI/The MIT Press, Menlo Park, CA, 1997.

[27] Elke Teich, Eli Hagen, Brigitte Grote, and John A. Bateman. From communicative context to speech: integrating dialogue processing, speech production and natural language generation. *Speech Communication*, 21(1–2):73–99, February 1997.

[28] David Traum and Staffan Larsson. The information state approach to dialogue management. In Ronnie Smith and Jan van Kuppevelt, editors, *Current and New Directions in Discourse and Dialogue*, pages 325–353. Kluwer Academic Publishers, Dordrecht, 2003.

[29] Terry Winograd and Fernando Flores. *Understanding computers and cognition: a new foundation for design*. Ablex, Norwood, New Jersey, 1986.

[30] Weiqun Xu, Bo Xu, Taiyi Huang, and Hairong Xia. Bridging the gap between dialogue management and dialogue models. In *Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*, pages 201–210, Philadelphia, USA, July 2002. Association for Computational Linguistics.