Dialogue Management in Natural Language Systems

Proceedings of the eleventh
Twente Workshop on Language Technology

Susann LuperFoy, Anton Nijholt, and Gert Veldhuijzen van Zanten (eds.)
CIP GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

LuperFoy, S., Nijholt, A., Veldhuijzen van Zanten, G.

Dialogue Management in Natural Language Systems
Proceedings Twente Workshop on Language Technology 11 / S. LuperFoy, A. Nijholt, G. Veldhuijzen van Zanten
Enschede, Universiteit Twente, Faculteit Informatica

ISSN 0929-0672

trefw.: natural language processing, speech, dialogue, dialogue management, language engineering

© Copyright 1996; Universiteit Twente, Enschede

Book orders:
University of Twente
Ch. Bijron
Dept. of Computer Science
PO Box 217
NL 7500 AE Enschede
fax: +31-53-4893503
Email: bijron@cs.utwente.nl

Druk- en bindwerk: Reprografie U.T. Service Centrum, Enschede
PREFACE

TWLT is an acronym of Twente Workshop(s) on Language Technology. These workshops on natural language theory and technology are organised by Project Parlevink (sometimes with the help of others), a language theory and technology project conducted at the Department of Computer Science of the University of Twente, Enschede, The Netherlands. Each workshop has proceedings containing the papers that were presented. For the contents of these proceedings consult the last pages of this volume.

Previous workshops.
TWLT9, *Corpus-based Approaches to Dialogue Modelling*, 9 June, 1995

TWLT 11, the present workshop, has been organized by a committee consisting of Harry Bunt, Susann LuperFoy, Anton Nijholt, Jan Schaaeke and Gert Veldhuijzen van Zanten.

This workshop focussed on the dialogue management task in natural language processing systems. Why did we want to organize a workshop on this particular theme? We feel that dialogue management is a key issue in natural language information systems. The management of a dialogue, however, is not a trivial problem. Several important questions come to mind: how can we monitor whether the dialogue is progressing efficiently? How much and what kind of feedback should the system give on what it has understood so far? When? How can the dialogue be controlled? What is the role of initiative in dialogues? These and many more issues must be addressed before we can hope to produce dialogue systems that operate in way that is acceptable to a large class of users. Therefore, dialogue management in natural language systems is an important scientific and technological issue.

The workshop was held in the 'Collegezaal-Complex' of the University of Twente. Just as with the previous workshop programs, there were presentations, including videos and demos, by a select group of internationally known researchers. The general aim was to provide a platform for the presentation of new developments and for the exchange of ideas between people working in the area of dialogue modelling.

A workshop is the concerted action of many people. We are grateful to the authors and the organizations they represent, for their efforts and contributions. But in addition we would like to mention here the people whose efforts have been less visible during the workshop proper, but whose contribution was evidently of crucial importance. Charlotte Byron, Alice Hoogvliet-Haverkate and Cindy Evers took care of the administrative tasks (registration, hotelreservations, etc.). The editors are most grateful to René Steetskamp who took care of all the papers and compiled them into these proceedings.

TWLT12, the next workshop in the series, will take place on 11-14 September 1996. It will also be presented as *International Workshop on Computational Humor* (IWCH). Its topic will be the automatic interpretation and generation of verbal humor. We hope it will match the success of this and the previous workshops.

Susann LuperFoy, Anton Nijholt and Gert Veldhuijzen van Zanten

June, 1996
CONTENTS

Workshop Papers:

Conversational Agency: The TRAINS-93 Dialogue Manager
David R. Traum (Université de Genève, CH) 1

Towards Multimodal Dialogue Management
Scott McGlashan (SICS, SW) 13

A Conversational Agent to Navigate in Virtual Worlds
Pierre Nugues, Christophe Godéreaux, Pierre-Olivier El Guedj & Frédéric Revolta
(GREYC, F) 23

Which Processes to Manage Human-Machine Dialogue?
Anne Vilnat (LIMSI-CNRS, F) 35

Tutoring versus Training: A Mediating Dialogue Manager for Spoken Language Systems
Susann LuperFoy (MITRE, USA) 45

Building on Experience: Managing Spoken Interaction through Library Subdialogues
David G. Novick and Stephen Sutton (Portland, USA) 51

Communicational Deviation in Finalized Informative Dialogue Management
Latifa Taleb (INRIA, F) 61

Speech Act Generation in Cooperative Dialogue
Robbert-Jan Beun (IPO, NL) 71

Pragmatic Interpretation and Dialogue Management in Spoken-Language Systems
Gert Veldhuijzen van Zanten (IPO, NL) 81

Topics in SCHISMA Dialogues
Joris Hulstijn, René Steetskamp, Hugo ter Doest (University of Twente, NL), Stan van de Burgt (KPN Research, NL) and Anton Nijholt (University of Twente, NL) 89

Dialogues in Air Traffic Control
Gavin Churcher, Clive Souter and Eric S. Atwell (Leeds University, UK) 101

Context Construction as Subtask of Dialogue Processing -- the VERBMOBIL Case
Elisabeth Maier (DFKI, D) 113

Dialogue Management in a Generic Dialogue System
Anders Baekgaard (CPK, DK) 123

Dialog Management in the CMU Spoken Language Systems Toolkit
Wayne Ward (Carnegie Mellon University, USA) 133

Understanding of Spontaneous Utterances in Human-Machine-Dialog
Wieland Eckert (University of Erlangen, D) 139

Some Ideas for the Automatic Acquisition of Dialogue Structure
Jan Alexandersson (DFKI, D) 149

Cooperative Response Planning in CDM: Reasoning about Communicative Strategies
Kristiina Jokinen (Nara Institute of Science and Technology, JP) 159
Dialogue Grounding for Speech Recognition Systems
Elizabeth Hinkelmann (Kurzweil Applied Science, USA)
169

Response Generation in Collaborative Dialogue Interactions
Jennifer Chu-Carroll (University of Delaware, USA)
177

Interaction Management Functions and Context Representation Requirements
Harry Bunt (Tilburg University, NL)
187

Dialogue Management in a Mixed-Initiative, Cooperative, Spoken Language System
Sandra Williams (BT, GB)
199

The Dialog Component in the Waxholm System
Rolf Carlson (KTH, SW)
209

Special Session
Evaluation of Spoken Dialogue Systems
Laila Dybkjær, Niels Ole Bernsen and Hans Dybkjær (Roskilde University, Denmark)
Sponsors and Support

We gratefully acknowledge help from:

University of Twente

KPN Research, Leidschendam

Prioriteitsprogramma Taal- en Spraaktechnologie, NWO
Hebben we over vijf jaar allemaal beeldtelefoon? Halen we straks massaal intelligente netwerken in huis? Zit er toekomst in spraaktechnologie? Of in cyberspace?

KPN Research heeft een naam hoog te houden als het gaat om technologische vernieuwing. In samenwerking met de voornaamste opdrachtgevers PTT Post en PTT Telecom, verricht zij onderzoeks- en ontwikkelingswerk, met als doel de dienstverlening van Koninklijke PTT Nederland (KPN) te verbeteren.

**Voorop in de markt**

KPN heeft zich tot doel gesteld om in 2000 internationaal gezien tot de grootste aanbieders op haar markt te behoren.

Een ambitieus plan, waarin KPN Research een belangrijke rol speelt.

De laatste ontwikkelingen in zaken als multimedia, Internet, intelligente netwerken, mobiele communicatie, elektromagnetisme, workflow management en de beveiliging van netwerken en diensten, zijn hier dan ook aan de orde van de dag.

Bouwen aan je toekomst. Èn die van KPN.

Het standaard carrièrepad bestaat niet bij KPN Research. Academici krijgen vrijheid en verantwoordelijkheid om hun eigen toekomst te sturen. Èn die van KPN.

Eigen initiatief, visie, een klantgerichte manier van werken, verantwoordelijkheidsgevoel en een sterke drive zijn essentieel om verder te komen in onze vestiging Leidschendam (350 medewerkers) of Groningen (100 medewerkers). Een universitaire studie technische natuurkunde, elektrotechniek, wiskunde, economie of (bedrijfskundige) informatica vormt een prima uitgangspunt. Daarnaast zijn je analytisch vermogen, je sociale vaardigheden en je motivatie belangrijke selectiecriteria.

**Informeren of solliciteren**

Wie meer wil weten, kan contact opnemen met de afdeling Personeel & Organisatie, telefoon 050 - 582 11 29. De Internet site van KPN Research bekijken, is natuurlijk ook mogelijk: “http://www.iway.nl/kpn_research.html”

Je kunt je schriftelijke sollicitatie met c.v. richten aan KPN Research, t.a.v. de heer J. Hasenoot, Postbus 15000, 9700 CD Groningen. Hij is ook te bereiken via email: J.Hasenoot@research.kpn.com.
CONVERSATIONAL AGENCY:
THE TRAINS-93 DIALOGUE MANAGER

David R. Traum
TECFA, FPSE
Université de Genève
9 Rte de Drize, Bat D
1227 Carouge, Switzerland.
David.Traum@tecfa.unige.ch

ABSTRACT

Designing an agent to participate in natural conversation requires more than just adapting a standard agent model to perceive and produce language. In particular, the model must be augmented with social attitudes (including mutual belief, shared plans, and obligations) and a notion of discourse context. The dialogue manager of the TRAINS-93 NL conversation system embodies such an augmented theory of agency. This paper focuses on the representation of mental state and discourse context and the deliberation strategies used in the agent model of the dialogue manager.

1 INTRODUCTION

A dialogue manager is that part of a dialogue system that connects the I/O devices and translators (whether they be spoken or typed language, a command language, menu selection, graphical presentation, etc.) to the parts that do the domain task reasoning and performance. In a simple language front-end system (e.g., for querying a database), dialogue management can be little more than a transducer from the I/O language to the task command language. However, for actual dialogue, this will also require some sensitivity to dialogue context – both for more flexible interpretation and more appropriate reaction. Figure 1 illustrates the connection of a dialogue manager to the rest of the system (in particular, the TRAINS-93 dialogue manager, described below in Section 4).

There are two main views of dialogue systems (and AI programs in general). One is as a tool designed for a particular task, the other is as an agent, with its own mental state. In the case of dialogue systems, the tool view is usually as a front-end interface to a task module with which the user would like to engage in a more flexible interaction.

![Diagram](image)

Figure 1: The TRAINS-93 System Architecture from a Dialogue Management Perspective

One claim of this paper is that it is more beneficial to design a dialogue system as an agent than as a front-end interface program. Since knowledge of many aspects of the domain task will be necessary even for accurate language interpretation, and dialogue structure is closely linked to task structure [Grosz and Sidner, 1986], for complex tasks (requiring more than a single interactional exchange to complete) a dialogue manager can not properly do it's job without at least an abstract notion of the internals of the task processing. Although the architecture in Figure 1 may look like that of a front-end system,
this is misleading, since the NL interpretation modules will need to consult with the dialogue manager and the domain task modules in order to get a full contextual interpretation of the utterances, while the dialogue manager will be responsible for the manner in which information produced by the domain specialists gets reported back to the user, following the conventions of natural language conversation.

In the next section, we outline some previous work on computational agents. This is followed by a discussion of how these models of agency must be extended to handle dialogue. In section 4, an example conversational agent is presented — the TRAINS-93 dialogue manager.

2 ARTIFICIAL AGENCY

There are currently many notions of agency, as discussed in [Wooldridge and Jennings, 1995]. For this paper, we consider an agent to be an entity with its own mental state and capacity for autonomous action. People are obviously prime examples of agents, however it can also be beneficial to view other entities (such as corporations or computer systems) as agents as well, as long as similar concepts can be fruitfully used to analyze their (past and future) behavior. McCarthy has a nice characterization of when it is useful to ascribe mental qualities to machines [McCarthy, 1990]. In particular, while some of the more emotional or particularly human qualities may not be useful, other qualities, such as beliefs and intentions can provide compact and easy to understand descriptions of the functioning of machines, as well as people.

Given that one can use mentalistic notions to describe machines, the important questions become which mental attitudes are most appropriate, how are they modeled, and how do they interact? [Wooldridge and Jennings, 1995] note that most theories include at least one informational attitude (such as knowledge or belief) and one pro-attitude (such as desires, goals, or intentions).

One popular formulation is the , BDI model, which include Beliefs, Desires, and Intentions as the primary mental attitudes. Figure 2 shows a schematic of such an architecture, simplified from [Bratman et al., 1988]. In this figure, the attitudes are shown as ovals, while the processes are shown as boxes. Arrows represent the flow of influence — inputs for the

processes and changes for the attitudes. The beliefs are the agent's model of the world — including both the current state, and how things have been in the past and what they are likely to be like in the future, given the performance of particular actions. By reasoning about its beliefs, the agent can derive new beliefs. Desires represent how the agent would like the world to be in the future. These will be the ultimate motive force in making the agent be more than just a passive observer of the world. The deliberation process will be one of deciding which actions should be performed so as to best achieve the desires. This process must consider the desires themselves, beliefs about how the world is now, beliefs about what kinds of actions are possible and what effects they achieve, as well as reasoning about the effects of the totality of actions under consideration.

The outcome of this deliberation process will be structured plans which the agent has decided to perform. These are the Intentions of the agent. According to Bratman's theory of intention [Bratman, 1987], these will play several important roles — most centrally, they contribute to the agent actually performing the action. Also, they will constrain future deliberation of an agent so as not to form conflicting intentions (if a conflicting desire is judged to be more important, a prior intention can be dropped). Also, the agent will monitor the success of actions and the achievability of intentions, replanning when needed in order to achieve the intentions. Some intentions which cannot be performed directly may also require further planning and deliberation (and adoption of intention) in order to be achieved.

The agent interacts with the world by performing actions and perceiving aspects of the world, including changes which result from these actions. Perceptions will influence the beliefs of the agent, while actions may change aspects of the world.

This general framework illustrates how mental at-


desire

Intensions

Beliefs

Perception

Deliberation

Reasoning

Action

Figure 2: BDI Agent Architecture

The original figure included separate boxes for means-end reasoning (i.e., planning), and a plan library — here we include this as part of the more general beliefs and reasoning. Also, Bratman, Pollack, and Israel included separate boxes for an opportunity analyzer and a filtering process, which here are included as part of the deliberation.

1
titudes and processes can be used to characterize the behavior of an agent. In [Bratman et al., 1988], the focus is on the role of resource bounds on various aspects of the deliberation process. This architecture was also tested and found to perform adequately in the Tileworld domain [Pollack and Ringouette, 1990]. Others have attempted to formalize these attitudes and their relations, e.g., Rao and Georgeff have used a branching time semantics to axiomatize some of the relations of these attitudes.

3 CONVERSATIONAL AGENCY

While the BDI model has been useful for modeling a solitary agent, the question arises as its adequacy for handling multi-agent phenomena, and specifically dialogue. As a first attempt, one could view the perception from Figure 2 as a process of natural language interpretation, and the action as natural language generation. Austin introduced the term *speech acts* to describe the view of language as performance of actions [Austin, 1962]. Work by Allen, Cohen, and Perrault (e.g., [Cohen, 1978, Cohen and Perrault, 1979, Perrault and Allen, 1980, Allen, 1983]) has brought this view within the computational community, by expressing these speech acts within AI theories of actions, with preconditions and effects relating to the mental states of the agents.

There are still several shortcomings with the BDI model of Figure 2, applied to conversation. These can be illustrated by a closer examination of AI planning operators for speech acts. Figure 3 shows one formulation of a request speech act, in which a speaker requests that a hearer perform some action. The body field represents a decomposition of the action into primitives; any action (or combination of actions) which achieve this body will count as performing a request action. In particular, the surface-request act which has this condition as its effect will be a request. A surface request, according to this formulation, is the utterance any imperative sentence, such as “Please make Orange Juice.” Imperative sentences (surface-requests) are not the only way to perform requests, however. Utterance of another sentence, such as “We need to make Orange Juice”, can also be a request, as long as it has the body conditions as an effect.

Getting back to Figure 2, we can see that several of the necessary aspects of the operators in Figure 3 are present, although some are missing. Most basically, a deficiency with Figure 2, even as a solitary agent model is that the arrow from perceptions affects only beliefs. More generally, perceptions can affect desires and other attitudes as well as beliefs. This can be seen in the effects of the request act in Figure 3. A second problem is that Figure 2 shows only the agent and the world. Yet, for conversation, there are at least two agents involved (speaker and hearer in Figure 3). A model of conversational agency must include a model of the conversational partner as well as of the deliberating agent and the world. As illustrated by the operators in Figure 3, this must include beliefs about the other agent’s mental state, as well as desires (or other pro-attitudes) that the other agent perform some action (such as a conversational response). In addition to simple nested attitudes about another agent, we claim that social attitudes, which link multiple agents, are also necessary to accurately model dialogue phenomena. Mutual Belief (MB) is one popular attitude, as illustrated in Figure 3.

A final missing ingredient for dialogue modeling is a notion of discourse context. Context will play an important part in deciding which of the possible acts are actually being performed. This context will include aspects of the mental state but will also include other aspects of the interaction, including the sequential patterns of action between the agents. In the rest of this section, we will discuss our own work in extending models of agency to be suitable for conversation, whereas dialogue context will be brought up again in the following section.

3.1 DISCOURSE OBLIGATIONS

Although Figure 3 correctly shows that attitudes other than belief must also be affected by action and perception, we have claimed that this desire of the hearer is proper effect of a request [Traum and Allen, 1994]. To examine this issue in more detail, we consider speech act plans from a related formalism [Cohen and Perrault, 1979]. Figure 4, from shows the planning process that an agent might use to decide to issue a request, given the desire for some action to be performed. Here, one agent, S, wants another agent, John, to perform some action α. S realizes that if John has a desire to perform α, then John may perform it.² Now,

²In the Cohen and Perrault formalism, this link is called a want precondition. Using the BDI model, however, we can assume that
S must find and perform some action that will have as an effect that John wants to perform $\alpha$. Here, an assumption of cooperativity is assumed, such that if John realizes that $S$ wants him to do $\alpha$, John himself will want to do $\alpha$, as well. And it is assumed that the direct effect of the request is a belief that $S$ wants John to perform $\alpha$. The upshot is that $S$ can perform this request and then rely on John's rational processes of perception, reasoning, and deliberation to intend and perform $\alpha$.

$$\alpha \ (\text{JOHN})$$

\[ \text{want.pr} \]

\[ \text{JOHN WANT } \alpha \ (\text{JOHN}) \]

\[ \text{effect} \]

\[ \text{CAUSE-TO-WANT(S,JOHN, } \alpha \ (\text{JOHN}) \]  

\[ \text{cando.pr} \]

\[ \text{JOHN BELIEVE S WANT } \alpha \ (\text{JOHN}) \]

\[ \text{effect} \]

\[ \text{REQUEST(S,JOHN, } \alpha \ (\text{JOHN}) \]

Figure 4: Cohen & Perrault (79) Plan for Request

This model has a wide range of applicability in cooperative situations, and has also been used to explain why an answer follows a question (i.e., if $\alpha$ is the performance of some speech act). This account is missing something important, however. First, it associates a request with any discovery of a desire by another agent. While an agent can use indirect means to perform a request, intuitively there is a difference between helpfully acting to fulfill a discovered desire and performing an action that has actually been requested; informing of a desire is not always requesting that the desire be met. More crucially, consider the cases in which an agent is not disposed to be cooperative. Here, according to the plan in Figure 4, the cause-to-want action will be blocked, and John might not perform $\alpha$. Also, even if the agent is disposed to be cooperative, perhaps he does not have the ability to perform $\alpha$. As a third case, the agent might have some prior goal or intention not to do $\alpha$. While this is fine, as far as it goes, in conversation, the agent will generally respond with something, even if it is not the desired action. What is it that inspires the agent to respond in these cases?

We claim that the more direct effect of a question, and the motivating factor in both the cooperative and non-cooperative setting is an obligation to respond. Obligations represent what an agent should do, according to some set of norms. The notion of obligation is a topic of much study, with some aspects are formalized as Deontic Logic (von Wright, 1951, McCarthy, 1990). These logics allow one to define permissible, obligatory, and forbidden actions. Just because an action is obligatory with respect to a set of rules does not mean that an agent will actually perform the action. So we do not adopt the social agent model suggested by [Shoham and Tennenholtz, 1992] in which agents' behavior cannot violate the defined social laws. If an obligation is not satisfied, then this means that one of the rules must have been broken. We assume that agents generally plan their actions to violate as few rules as possible, and so obligated actions will usually occur. But when they directly conflict with the agent's personal goals, the agent may choose to violate them. Obligations are quite different from and can not be reduced to intentions and goals. In particular, an agent may be obliged to do an action that is contrary to his goals. Obligations thus form a dual to desires as motivating inputs to the deliberation procedure that leads to the adoption of intentions. An agent must consider both in deciding which actions to perform and in setting priorities for this action.

Specific obligations arise from a variety of sources. In a conversational setting, an accepted offer or a promise will incur an obligation. Also, a command or request by the other party will bring about an obligation to perform the requested action. If the obligation is to say something then we call this a discourse obligation. Our model of obligation is very simple. We use a set of rules that encode discourse conventions. Whenever a new conversation act is determined to have been performed, then any future action that can be inferred from the conventional rules becomes an obligation. We use a simple forward chaining technique to introduce obligations.

Some obligation rules based on the performance of conversation acts are summarized in Table 1. When an agent performs a promise to perform an action, or performs an acceptance of a suggestion or request by another agent to perform an action, the agent obliges itself to achieve the action in question. When another agent requests that some action be performed, the request itself brings an obligation to address the request: that is, either to accept it or to reject it (and make the decision known to the requestee) - the requestee is not permitted to ignore the request. A question establishes an obligation to answer the question. If an utterance has not been understood, or is believed to be deficient in some way, this brings about an obligation to repair the utterance.
source of obligation | obliged action
---|---
$S_1$ Accept or Promise A | $S_1$ achieve A
$S_1$ Request A | $S_2$ address Request: accept or reject A
$S_1$ YNQ whether P | $S_2$ Answer-if P
$S_1$ WHQ P(x) | $S_2$ Inform-ref x
utterance not understood or incorrect | repair utterance
$S_1$ Initiate DU | $S_2$ acknowledge DU
Request Repair of P | Repair P
Request Acknowledgement of P | acknowledge P

Table 1: Sample Obligation Rules

The model of obligations as the main effect of requests leads to both broader coverage and a more direct planning and deliberation procedure than the Cohen and Perrault model of Figure 4. Our model of the cooperative case is shown in Figure 5. Here, the obligation is the direct result, which feeds into the deliberation process. If the agent John is cooperative, without conflicting desires, then as before, the deliberative process will lead to the adoption of an intention to perform $\alpha$. In the case in which there is some conflicting desire, the obligation to respond still motivates some action, in this case some other $\beta$, such as an evasion or refusal.

$$\alpha(\text{JOHN})$$

$$\text{JOHN INTEND } \alpha(\text{JOHN})$$

$$\text{Deliberation}$$

$$\text{OBLIGED(\text{JOHN, S, ADDRESS REQUEST(...))}}$$

$$\text{effect}$$

$$\text{REQUEST(S,JOHN, } \alpha(\text{JOHN}))$$

Figure 5: Traum & Allen (94) Model of Requests

3.2 Mutual Belief and Grounding

Another deficiency of the speech act model in Figure 3 (and in fact almost all prior speech act formalisms in dialogue systems) is the way mutual belief (MB) is assumed to be the direct result of the utterance of a sentence, such as an imperative. In fact, examining actual conversation reveals that there is an elaborate process of feedback that can accompany initial utterances, and it is generally only after some sort of acknowledgment that the assumption of mutual belief is made. One reason for this is that both the language production and language interpretation are error-prone processes. A speaker cannot assume with any confidence that her contribution has been understood without some feedback from her interlocutor. Lack of understanding can be signaled with some sort of repair or request for repair. In cases in which the speaker does not receive any feedback, one can observe requests for acknowledgment or repetitions and refashionings of the original contribution in an attempt to elicit some kind of feedback. As an example of the prevalence of this kind of feedback in spoken dialogue, we observed that over half of the speaker transitions in a task-oriented dialogue corpus commenced with a simple acknowledgment, such as "ok", whereas another 30% provided direct evidence of the level of understanding by continuing with related content, while 15% followed turns consisting only of such acknowledgments. Only 2-5% of turns transitions commenced with material unrelated to the previous turn [Traum and Heeman, 1996].

Clark and Schaefer call this process of reaching mutual belief (or common ground) grounding [Clark and Schaefer, 1989]. They present a descriptive model, in terms of presentation and acceptance phases that allow them to track the augmentation of common ground as the conversation proceeds. Their model is not well-suited for an on-line agent involved in dialogue, however, since it requires examination of subsequent spans of text in order to determine the boundaries of these phases.

We have built on this work, adapting it to something more useful for an on-line agent by presenting a speech acts approach to grounding, in which utterances are seen as actions affecting the state of grounding of contributed material [Traum and Allen, 1992, Traum, 1994]. In addition to some acts which present new material, there are acknowledgment acts which signal that the current speaker has understood previous material presented by the other speaker, repairs and requests for repair.

While there are short-comings of previous accounts of speech acts such as that in Figure 3, particularly in the way content is added (or is not added) to mutual
belief, we prefer to keep as much of the previous theory intact as possible. We keep all of the core speech acts of previous work, recognizing, however, that they are really multi-agent actions, which require input from multiple participants in order to have their full effects such as mutual beliefs. We introduced a level of dialogue structure called discourse units (DUs), at which these core speech acts are completed. These DUs are built up by single-utterance grounding acts. Recognizing the fact that multiple types of action occur in conversation, we extended speech act theory to the multi-level conversation act theory, summarized in Table 2. As well as the grounding and core speech acts, there are also levels to model turn-taking behavior and higher order coherence of dialogue. A similar model of meta-locutionary acts was previously introduced by Novick [1988].

<table>
<thead>
<tr>
<th>Level</th>
<th>Act Type</th>
<th>Sample Acts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;DU</td>
<td>Turn-taking</td>
<td>take-turn, keep-turn</td>
</tr>
<tr>
<td>UU</td>
<td>Grounding</td>
<td>Initiate Repair, Ack Continue</td>
</tr>
<tr>
<td>DU</td>
<td>Core Speech Acts</td>
<td>Inform Y/N, Accept Request</td>
</tr>
<tr>
<td>&gt;DU</td>
<td>Argumentation</td>
<td>Elaborate Q&amp;A</td>
</tr>
</tbody>
</table>

Table 2: Conversation Act Types

The grounding acts were used as the basis for a computational account of the grounding process [Traum, 1994], in which a finite automaton was used to track the state of a DU, given a sequence of grounding acts in conversation. This model could also be used to predict possible subsequent acts as well as determine which act(s) must be performed in order to have a grounded DU (which would thus realize the effects of the constituent core speech acts).

Also presented in [Traum, 1994] was an account of grounding in terms of the perception and mental attitudes of the conversational participants. This included beliefs, intentions, and obligations, as well as nested attitudes. The reasons for engaging in particular actions were presented as relations between elements of these mental states, along with a description of the effects of these actions on the mental states of conversing agents.

4 THE TRAINS-93 DIALOGUE MANAGER

The aspects of conversational agency described in the previous section have been put together to form the core of a theory of dialogue management, of which the TRAINS-93 dialogue manager is an example implementation. The TRAINS-93 System (described in detail in [Allen et al., 1993]) is a large integrated natural language conversation and plan reasoning system. The dialogue manager is responsible for maintaining the flow of conversation and making sure that the conversational goals are met. For this system, the main goal is that a shared plan which meets the user's domain goals is constructed and executed in the domain.

The dialogue manager must track the state of the dialogue, determine effects of observed conversation acts, generate utterances, and send commands to the domain task reasoners when appropriate. Each utterance will generally contain acts (or partial acts) at each of the conversation act levels.

As with the BDI agency model in Figure 2 the dialogue manager agent can be seen as composed of two main parts, a context representation, which includes various aspects of mental state, and a deliberation and action mechanism. The actor is reactive to the discourse context, looping to perform only small atomic actions in response to its current state, and then updating and reacting again. This architecture makes it fairly flexible to new observations of the domain or conversation from the user as well as to its other processing. In addition, the context is used to help determine which acts have been performed, and is updated with the results of those acts.

4.1 DISCOURSE CONTEXT

The discourse context of the dialogue manager contains representations of both the mental attitudes described previously, as well as a model of the current conversation. Several nestings of domain belief must be tracked in order to fulfill the conversational purposes in a task-oriented dialogue. Private beliefs must be maintained in order to do accurate task reasoning. In addition, the system must maintain beliefs about the user's beliefs and about mutual beliefs both to interpret user utterances and formulate its own expressions coherently. We represent these belief nestings as distinct but related belief modalities.

In addition to actual beliefs, the system must also track proposals. Although there will be a natural connection between these and actual beliefs, there is good reason to keep them distinct during intermediate phases in the conversation. One use is for modeling insincere utterances. Even if the interpretation of an utterance includes a claim that a certain state of affairs holds, there might be good evidence to suppose that the actual beliefs of a speaker are otherwise. Having separate modalities for proposals allows representa-
tion of any discrepancy. In addition, this will allow a more explicit representation of the method by which beliefs change through conversation — the immediate effect of a representational utterance will change only the proposal modality, and it will require an additional (mental) action to actually change the belief. This proposal modality is also useful for representing suggested courses of events that have not yet been firmly decided upon.

![Diagram of Belief and Proposal Contexts]

Figure 6: Belief and Proposal Contexts

Figure 6 shows the relationships between the belief and proposal modalities. The belief modalities are shown with solid boxes, with containment representing the relationships between these modalities. Some of these beliefs will concern proposals. Part of the mutual beliefs will be mutual beliefs about what the user and system have proposed. Proposals that have been accepted are also shown here as shared. For representative proposals (about the actual state of affairs), this shared modality will be undistinguished from general mutual beliefs about the world, but for directive proposals (about actions to be taken by the conversational participants), the shared context will include an intentional component (to perform these actions and keep them achievable) as well as a mutual belief about future eventualities. The proposal modalities in the Sys Bel User Bel context represent proposals that have been initiated but not yet grounded. Also, in the Sys Bel context, the System Private modality represents proposals that the system has decided to make but has not yet initiated.

For the TRAINS system, many of the utterances will include suggestions of actions, goals, and constraints to add to the current domain plan. From the point of view of the dialogue manager, domain plans are abstract entities which contain a number of parts. These include: the goals of the plan, the actions which are to be performed in executing the plan, objects used in the plan, and constraints on the execution of the plan. The detailed structure of TRAINS-93 domain plans and the view of plans as arguments are described in detail in [Ferguson, 1995].

Views of domain plans are represented as follows. The shared modality will include aspects of plans assumed to be shared plans — jointly intended by the system and the user. The mutually believed proposal modalities include plans proposed by one or the other party but which have not yet been accepted. The proposal modalities in Sys Bel User Bel represent proposals which have not yet even been acknowledged, and finally the System Private modality will contain plans that the system’s back-end plan reasoning has constructed but which have not yet been communicated to the user. This framework allows for the representation of both the incremental construction of plans as well as conflicting proposals of what a plan should be, when the plans in different contexts have contradictory components.

The system maintains a set of high-level Discourse Goals representing what it hopes to achieve with the conversation. For the TRAINS domain, this is represented as a script, specifying the goals of different phases in the conversation. The TRAINS-93 script includes phases for identifying a domain goal, developing a shared plan to meet this goal, and executing the plan in the TRAINS domain. The system also maintains structures of obligations which have arisen according to the rules in Table 1. Also, a set of intended conversation acts is maintained, which the system will try to perform (by sending to the NL Generator, for output to the user), when it gets an opportunity. Because the system attempts to adhere to conventional interactional patterns, it does not always perform these right away, and might not get a chance to perform some of them. For example a suggestion may be preempted by a similar or conflicting suggestion by the user. Also, an answer to a question may become irrelevant if the user retracts the question.

4.1.1 Discourse Model

In addition to the mental attitudes described above, there are several aspects of the discourse interactional state that must be tracked. These represent the model of the conversation itself rather than particular mental attitudes of the participants. They are used as a resource to help generate local expectations for the flow of the conversation, as well as serving as an important
tool for interpreting subsequent utterances.

Two contextual notions are useful for tracking turn-taking in casual task-oriented conversations such as those in the TRAINS domain. These are the turn and local initiative. Each of these may be said to be held by one (or none) of the participants at any given time in the conversation. The notion of who has the turn is important in deciding whether to wait for the other agent to speak, or whether to formulate an utterance. It will also shape the type of utterance that will be made, e.g. whether to use some kind of interrupting form or not.

Local initiative\(^3\) can be glossed as providing the answer to the question of who has the most recent discoursal obligation — who is expected to speak next according to the default plans for simplest satisfaction of conversational goals. In the TRAINS domain, the initiative is shared, with different participants holding it at different points in the conversation. In the initial phase, the user has the initiative while the task is conveyed. In the main part of the conversation — the construction of the plan — the initiative can lie with either party, though it generally remains with the user. In the final phase, verifying successful completion of the problem, the initiative belongs with the system.

In order to track grounding, the system will maintain a bounded stack of accessible discourse units (DUs). For each DU, the system notes the initiator and the state of the DU, in addition to a representation of the constituent (partial) core speech and argumentation acts and their effects. Multiple DUs are modeled using a bounded stack structure. The structure is stack-like since, generally, new utterances will affect the most recently started DU. The stack structure is bounded to capture the constraint that DUs have limited accessibility. After enough intervening material, the older DUs are no longer directly accessible (although their content can always be reintroduced in new DUs). Uncompleted DUs which “fall off” the back of the stack are treated as if they had been cancelled — their contents are not considered grounded.

Discourse Segmentation information [Grosz and Sidner, 1986] is kept for a variety of purposes in linguistic interpretation and generation, including the ability to determine the possible referents for a referring expression and the intentional relations between utterances. The currently open segment structure will guide how certain utterances will be interpreted. In addition to general segmentation information, a structure of conversationally accessible domain objects is maintained. For the TRAINS system, this will include a set of accessible domain plans from a given segment, as well as recency pointers to parts of plans from utterances comprising the segment.

4.2 Conversational Updates

The main way that the conversational state is updated is through the performance of conversation acts. These are briefly summarized here. Other changes to the mental state result from the deliberation process described in the next section. Turn-taking acts will generally only affect the turn. Grounding acts will primarily affect the grounding model — they will update the state of the DU they are a part of. Each grounding act performed as part of an utterance event will also have associated with it a (possibly empty) list of core speech acts and argumentation acts which are attempted in the utterance, which are also added to the local memory of that DU. Initiate and ack acts have additional consequences. An initiate will add a new DU to the DU stack, often removing an old DU from the bottom of the DU stack. An ack will make the contents of that DU mutually believed, thus causing a transfer of contents from the SBUB modality to the MB modality, perhaps causing additional effects such as new discourse obligations or intentions.

Core speech acts have a variety of effects, such as adding new mutual beliefs or obligations, as outlined above. More details are presented in [Allen et al., 1995, Traum, 19941]. Argumentation acts have an affect on the discourse coherence. This will result both in adding implicated information to the information conveyed by core speech acts, as well as affecting the discourse segmentation structure.

4.3 The Discourse Actor

In designing an agent to control the behavior of the dialogue manager, we choose a reactive approach in which the system will deliberate as little as possible until it can act in one way or another. It will not form complete or long-range plans about the discourse, but will proceed one step at a time, deducing and performing the next appropriate action, according to conversational conventions and its high-level goals. The TRAINS-93 actor uses the following prioritized sources for the deliberations:

1. Discourse Obligations from Table 1
2. Weak Obligation: Don’t interrupt user’s turn
3. Intended Speech Acts
4. Weak Obligation: Grounding
5. Discourse Goals: Proposal Negotiation
6. High-level Discourse Goals

\(^3\)Roughly the same notion as Control in [Walker and Whittaker, 1990], although we use a finer grained notion of utterance types.
The actor's first priority is fulfilling obligations. If there are any, then the actor will do what it thinks best to meet those obligations. If there is an obligation to address a request, the actor will evaluate whether the request is reasonable, and if so, accept it, otherwise reject it, or, if it does not have sufficient information to decide, attempt to clarify the parameters. In any case, part of meeting the obligation will be to form an intention to tell the user of the decision (e.g., the acceptance, rejection, or clarification). When this intention is acted upon and the utterance produced, the obligation will be discharged. Other obligation types are to repair an uninterpretable utterance or one in which the presuppositions are violated, or to answer a question. In question answering, the actor will query its beliefs and will answer depending on the result, which might be that the system does not know the answer.

In most cases, the actor will merely form the intention to produce the appropriate utterance, waiting for a chance, according to turn-taking conventions to actually generate the utterance. In certain cases, though, such as a repair, the system will actually try to take control of the turn and produce an utterance immediately. For motivations other than obligations, the system adopts a fairly "relaxed" conversational style; it does not try to take the turn until given it by the user unless the user pauses long enough that the conversation starts to lag. When the system does not have the turn (priority 2), the conversational state will still be updated, but the actor will not try to deliberate or act.

When the system does have the turn, the actor first (after checking obligations) examines its intended conversation acts (priority 3). If there are any, it calls the NL generator to produce an utterance. System utterances are also reinterpreted (as indicated in Figure 1) and the conversational state updated accordingly. This might, of course, end up in releasing the turn. It might not be convenient to generate all the intended acts in one utterance, in which case some intended acts may be left for the future consideration. When the turn changes, only those intended speech acts that are part of the same argumentation acts as those which are uttered will be maintained as intentions — others will revert back to whatever caused the intention to be formed, although subsequent deliberation might cause the intentions to be re-adopted.

If there are no intended conversation acts, the next thing the actor considers is the grounding situation (priority 4). The actor will try to make it mutually believed (or grounded) whether particular speech acts have been performed. This will involve acknowledging or repairing user utterances, as well as repairing and requesting acknowledgment of the system's own utterances. Generally, grounding is considered less urgent than acting based on communicative intentions, although some grounding acts will be performed on the basis of obligations which arise while interpreting prior utterances.

If all accessible utterances are grounded, the actor then considers the negotiation of domain beliefs and intentions, represented in Figure 6 (priority 5). The actor will try to work towards a shared domain plan, adding intentions to perform the appropriate speech acts, including accepting, rejecting, or requesting retraction of user proposals, requesting acceptance of or retracting system proposals, and initiating new system proposals or counterproposals. The actor will first look for User proposals which are not shared. If any of these are found, it will add an intention to accept the proposal, unless the proposal is deficient in some way (e.g., it will not help towards the goal or the system has already come up with a better alternative). In this latter case, the system will reject the user's proposal and present or argue for its own proposal. Next, the actor will look to see if any of its own proposals have not been accepted, requesting the user to accept them if they have been simply acknowledged, or retracting or reformulating them if they have already been rejected. Finally, the actor will check its private plans for any parts of the plan which have not yet been proposed. If it finds any here, it will adopt an intention to make a suggestion to the user.

If none of the more local conversational structure constraints described above require attention, then the actor will concern itself with its actual high-level goals (priority 6). For the TRAINS system, this will include making calls to the domain plan reasoner and domain executor, which will often return material to update the system's private view of the plan and initiate its own new proposals. It is also at this point that the actor will take control of the conversation, pursuing its own objectives rather than responding to those of the user.

Finally, if the system has no unmet goals that it can work towards achieving, it will hand the turn back to the user or try to end the conversation if it believes the user's goals have been met as well.

4.4 Example

The following example gives a small taste of how the dialogue manager uses this representation of context and priorities to engage in dialogue. More extended examples are presented in [Traum and Allen, 1994].
Traum, 1994]. The example starts with a declarative utterance by the User:

U: "There are oranges at Corning."

At the core speech act level, this is interpreted as performing both an inform (about the location of oranges), and a suggestion that the oranges be used in the current plan. At the grounding level, this is seen as the initiation of a DU. It is also seen as keeping the turn. This has the following effects on the context - first (at priority level 4), there is an unacknowledged DU, which will require grounding. More prominently, however, the user still has the turn, so the system will just wait for the next utterance.

U: "Is a boxcar there?"

This is interpreted as asking a yes-no question, continuing the current DU, and releasing the turn. Now there is an additional core speech act in the ungrounded DU, and the system has the turn. The chosen action is now, at priority 4, to add the intention to acknowledge the content in this DU (a new item at priority 5). Forming this intention also causes the system to update its mental state with the effects of this content. In this case, the inform and suggestion will lead to items in the user-proposed context in Figure 6, at priority level 5. The YNQ leads to an obligation to answer the question, which is at priority level 1. Since the obligation is of highest priority, the system acts upon this by querying its beliefs to see if a boxcar is at Corning. This check returns negatively, which leads the system to intend to inform the user of this fact. Now, the the highest priority are the intended speech acts. These are passed to the NL generator, and a combined, acknowledgment/answer is provided with:

S: "No there isn’t"

This simple example displays some of the flexibility of the reactive agency model. Given different responses or a different initial mental state, many variants of this simple dialogue could have been produced using the same rules. Most of the flexibility of plan-based approaches is maintained, while the obligation model presents a much more direct account of question answering, without any need for reasoning about or adopting the desires of the user.

5 CONCLUSIONS

The model presented here allows naturally for mixed-initiative conversation and varying levels of cooperativity. Following the initiative of the other can be seen as an obligation-driven process, while leading the conversation will be goal-driven. Representing both obligations and goals explicitly allows the system to naturally shift from one mode to the other. In a strongly cooperative domain, such as TRAINS, the system can subordinate working on its own goals to locally working on concerns of the user, without necessarily having to have any shared discourse plan. In less cooperative situations, the same architecture will allow a system to still adhere to the conversational conventions, but respond in different ways, perhaps rejecting proposals and refusing to answer questions. This architecture can handle production and recognition of acknowledgment and repair in a natural and fairly comprehensive manner, and can prompt for them when they are required. While the strict prioritization of aspects of mental state used here is too severe in general (e.g., some goals should take priority over some obligations), the framework still has a great deal of flexibility, seizing the initiative when it lags, and relinquishing it when the user presents her own goals.

Viewing dialogue systems, and in particular dialogue managers as agents naturally lends itself to such flexible interaction. Modeling the system’s action and computation as deliberation over aspects of mental state has several advantages. First, it allows a uniform treatment of dialogue partners, be they humans or other machines. The system can reason about itself in the same way it reasons about human users. Since much of the semantics and pragmatics of natural language communication makes reference (both explicitly and implicitly) to the mental state of communicating agents, having such an explicit model makes reasoning about natural language more straightforward.

Conversation and social interaction puts extra demands on a model of agency, beyond that of simple perception. The inclusion of social attitudes, such as obligation and mutual belief, however, can lead to natural and powerful extensions to a BDI model. As demonstrated by the TRAINS-93 system, the combination of a rich notion of mental state and simple, reactive deliberation mechanisms can yield flexible and dynamic conversational behavior.

REFERENCES


TOWARDS MULTIMODAL DIALOGUE MANAGEMENT

Scott McGlashan
Swedish Institute of Computer Science
Box 1263, S-164 28 Kista, Sweden
E-mail scott@sics.se

abstract

Effective dialogue management is a key issue in speech-based interfaces to information systems since it can ensure a cooperative interaction with the user. Cooperativeness requires techniques which allow the user to efficiently access information and also techniques which compensate for limitations in system knowledge and speech technology. The paper describes management techniques developed in a speech-only dialogue system and how they are being extended for a multimodal system which combines a direct manipulation interface with a spoken dialogue interface for a simple consumer information service.

1 introduction

This paper describes dialogue management techniques which have been developed for spoken dialogue systems, and how these techniques are being used in a multimodal system which combines a speech interface with a graphical user interface. Our guiding principle is that dialogue is a joint activity in which user input is interpreted as instructions how to update the system's model of the evolving dialogue, and that system output transparently reflects the state of this model. The model itself encodes techniques for contextual semantic interpretation, and dialogue strategies to handle clarification, confirmation and failure-repair in manner appropriate to dialogue progression. The approach is being extended to incorporate non-speech input and output, references to visual objects, output modality selection, and adaptive navigation. This will allow us to investigate the tension between speech and graphical modalities in a system providing a simple consumer information service.

2 spoken dialogue management

Over the last five years, spoken language dialogue systems have emerged for a variety of task domains including spatial navigation (Voyager), travel planning (Waxholm), and speech translation (Verbmbil). One particular area of interest has been in telephone-based dialogue systems which provide access to simple information services such as train timetable and flight enquiries (Peckham 1993; Aust and Oerder 1995). In these systems, "the customer or client identify certain entities to the person providing the service; these entities are parameters of the service, and once they are identified the service can be provided" (Hayes and Reddy 1983: 252). A simple dialogue from the domain of flight enquiries illustrates this type of service.

(1) S1: Welcome to British Airways flight enquiries service. How can I help you?
U1: Can you tell me the arrival time of BA777 from Stockholm?
S2: BA777 from Stockholm?
U2: Yes.
S3: BA777 leaves Stockholm at 11.17 and arrives London Heathrow terminal 1 at 13.41. That's BA777 arriving London Heathrow terminal 1 at 13.41. Do you have another enquiry?
U3: No.
S3: Thank you for calling. Goodbye.

The user provides parameters (the flight identifier, departure city and a request for the arrival time) and the system then provides the flight enquiry service.

These dialogue systems are inevitably compared by users to human agents offering a similar service. By analyzing human-human and (simulated) human-computer dialogues, we find that while users expect reduced linguistic competence from a system (and reflect this in
a simplification of their own linguistic behaviour), they still expect the system to retain many of the characteristics of a human service agent, whilst compensating for its own performance limitations (Giacin and McGlashan 1996; Wooffitt et al. forthcoming). Some of the main dialogue characteristics with systems include:

**global structure** the dialogue has an opening, a body and a closing. In the body, the system takes responsibility for obtaining information

**mixed initiative** while the user generally takes the initiative (such as providing task information or asking for repetition), the system can take the initiative to confirm information has been correctly understood, obtain information necessary for database access and, if the dialogue deteriorates, to constrain the form and content of user utterances (for example, by means of closed questions)

**over-informativeness** users may provide over-informative answers; for example, instead of U2, the user might have added the departure time as with *Yes, it left Stockholm late morning*

**contextual interpretation** user input may only provide partial and ambiguous information whose interpretation needs to be established in the discourse context; for example the utterance, *from Stockholm*, would function as a repetition in U2 of (1), but as a modification following *BA777 from Scunthorpe?*

**failure-repair** users expect the system to repair failures which may arise from performance limitations, especially speech recognition, as well as limitations in system knowledge — linguistic knowledge, semantic knowledge, task knowledge, and dialogue knowledge. The system needs to adopt pre-emptive strategies to avoid dialogue failure — such as confirming task information as in S2 — and reactive strategies to deal with failures when they do arise — such as asking for parameters to be spelt.

In the following sections we describe our techniques for addressing these characteristics. Other information services may exhibit different dialogue characteristics and so may require other dialogue management techniques (Bernsen et al. 1994).

### 2.1 basic principles

A minimal requirement is that the system plays the role of a co-operative agent so that the resulting interaction is comfortable, comprehensive and comprehensible to the user. From the user's point of view, whether the interaction is co-operative or not is judged solely on the basis of what the system says. Even if difficulties arise in the interaction, the system should still provide a response which does not lead to dialogue failure: while the goal of the interaction — providing the service — may fail, the dialogue *per se* should not (McGlashan et al. 1992). To achieve this, both interpretation of user utterances and production of system utterances must be informed by past and current states of the interaction. Co-operative dialogue management, therefore, requires the construction and maintenance of an interactional model: i.e. a model which specifies the layers of structure which can be distinguished in dialogue interactions. We distinguish linguistic structure, attentional (or discourse) structure, and intentional structure. Intentional structure is further differentiated into dialogue structure and task structure (Bunt 1989).

Of these structural layers, the characterization of the intentional is the most contentious. Three approaches have been distinguished (Cohen 1995): those based on dialogue grammars, those based on plans and intentions, and those, like ours, which treat dialogue as a joint activity where cooperating agents evolve a common model of the discourse situation. Our approach is based on the following general principles (Giacin and McGlashan 1996; Heisterkamp and McGlashan 1996):

1. Only the system's goals are explicitly represented in the dialogue model: user utterances are not assigned dialogue acts.
2. Only local transitions are modelled: the dialogue as a whole is not modelled, but global structure can still emerge.
3. Task-level information in user utterances is assigned a semantic function indicating its effects on the accessible part of the discourse model. Pragmatic functions are assigned on the basis of surface properties, including discourse markers.
4. These functions are applied to goals in the current dialogue model: they may satisfy a goal, modify it, or introduce another one. The results are then evaluated to determine which
goals provides an optimal continuation of the dialogue.

5. The system reports these goals to the user. The user is thus able to verify the system's model against their own interactional model. If verification fails, the user has an opportunity to make the problem explicit and correct it, so forestalling more serious problems which lead to irreparable breakdowns in the dialogue.

This approach is realized in a dialogue manager where each structural layer is represented in a separate model and each model, together with maintenance and update routines, is encapsulated in a semi-autonomous software module: a linguistic interface, semantics module, task module, and dialogue module. When the dialogue manager is ready to process user input, the linguistic interface calls the parser with a set of predictions, and communicates the result of recognition and parsing to the dialogue module. The parser result is either a semantic representation of the user turn, or an error message. After semantic, task and dialogue processing, a set of goals is selected as the dialogue continuation. The linguistic interface passes these to a linguistic generator for linguistic processing and synthesis.

2.2 semantic techniques

The primary function of the semantics module is to interpret the representation of user turns with respect to the discourse model and assign semantic functions to each task parameter in the user input.

Input from two types of parsers is supported: those which provide a compositional semantic representation, such as Unificaction Categorial Grammar; and those which produce frame-based semantic representation, such as phrase-spotting and finite-state grammar approaches. Compositional semantic input is reduced to a frame representation by application of inference rules: assignment of semantic function (as well as database access) in simple information services only requires a limited set of concepts extracted from user input (cf. translation applications). The frame representation consists of instantiated concepts (with 'modus' features such as definiteness and number) organized in an inheritance hierarchy. The hierarchy includes a meta-level and an object-level: the former describe information about the dialogue including discourse-level actions such repeat and open; while the latter principally refers to actions and concepts (such as request and flight) in the task domain. The representation for Can you tell me the arrival time of BA777 from Stockholm is illustrated in (2).

\[
\left( \begin{array}{c}
\text{type : request} \\
\text{type : flight} \\
\text{flightid : ba777} \\
\text{sourcecity : stockholm} \\
\text{goalttime ?!}
\end{array} \right)
\]

Analysis of the turn consists of a single object-level representation of the type request whose value is a flight concept with two instantiated parameters and one requested parameter.

Object-level descriptions are then added to the discourse model and their semantic functions assigned; meta-level descriptions can be directly given over to dialogue interpretation since their pragmatic function is already specified. The discourse model is composed of an ordered set of discourse states. Each state is defined as a structure of the type

\[ ID \times OWNER \times TYPE \times OBJECT \times PARAMETERS \times FUNCTIONS \]

where TYPE, OBJECT and PARAMETERS are extracted from the description and OWNER indicates whether the utterance was produced by the system or user. These states are ordered by recency. The semantics functions are then assigned by parameter-wise comparison with the most accessible, compatible concept; accessibility is simply based on recency, and compatibility depends on the type and modus properties of concepts. The set of semantic functions is shown in Table 1.

<table>
<thead>
<tr>
<th>Function</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>a new parameter has been introduced by the user</td>
</tr>
<tr>
<td>modified</td>
<td>the user has given an alternative value for an existing parameter</td>
</tr>
<tr>
<td>repeated</td>
<td>the user repeated a value for an existing parameter</td>
</tr>
<tr>
<td>negated</td>
<td>the user has negated the value of an existing parameter</td>
</tr>
<tr>
<td>requested</td>
<td>the user has requested the value of a parameter</td>
</tr>
<tr>
<td>inferred</td>
<td>the system has inferred a parameter value</td>
</tr>
</tbody>
</table>

Table 1: Semantic Function Assignments
With U1 in (1), there is no existing object of the same type, so a new id value flight1 is instantiated and the parameters flightid and sourcecity are assigned the function new, while goaltime is assigned the function requested. A more interesting situation arises if, instead of U2, the user provides an over-informative response like Yes, BA777 from Scunthorpe leaving late morning where the system has misrecognized the user’s repetition of Stock- holm as Scunthorpe and the user also provides the departure time. The following discourse state would be generated:

(3)  
```
  id: d4
  owner: user
  type: inform
  object: [id: flight1, type: flight]
  parameters: [flightid: ba777, sourcecity: scunthorpe, source: [1000, 1200]]
  functions: [flightid: repeated, sourcecity: modified, source: inferred]
```

The description is compatible with an existing object flight1: while the flight identifiers are the same, the value of the sourcecity parameter has been modified, and the user has introduced a new parameter source:time whose value has been inferred as a time interval.

Comparison with an accessible, compatible object may also resolve partial descriptions (see Section 4.2 below) as well as underspecified parameters. For example, if the user replies to a system request for the departure time with the utterance 11.20, comparison with the representation of the system utterance indicates that this time parameter needs to be contextually interpreted as a source:time parameter. Finally, since a user turn may consist of one or more utterances — either because that is how it was uttered or because recognition errors, speech disfluencies, out of vocabulary items, or gaps in the recognition grammars lead to ‘fragmentary input’ — the semantic interpretation mechanism updates the discourse model on an utterance-by-utterance basis. In this way, the representation of each utterance can be compared with the interpretation resulting from interpretation of the previous utterance in the turn. The effect is that task information in multi-utterance turns are assigned the same functions as in single utterance turns, except that a set of interpretations will be passed onto the dialogue interpretation function.

2.3 task techniques

Task parameter values are passed to the task module for updating its model. Since task structure determines many dialogue continuations, the task module embodies navigation strategies to efficiently obtain information necessary for successful database access, as well as techniques for suggesting alternative solutions and presenting the information to the user. These strategies result in goals being forwarded to the dialogue module.

The task module checks whether the task model is sufficiently instantiated for database access. This is determined by matching the task model against a set of request templates. Each template specifies obligatory parameters for a particular type of request. For example, a flight enquiry about the arrival time uses the following template:

(4)  
```
  input: goaltime
  required: {flightid, date, sourcecity, goalcity, source:time, [date, source:time], goal:time, goalcity, goalterminal, goaltime}
  output: goaltime
```

The template can be satisfied with either the flight number and date, or the departure and arrival cities, the date, and the departure time of the flight. If the task model does not completely match the request template, then one of the required parameters is sought from the user. In dialogue (1), the flightid is provided by the user but the date is inferred. Default constraints are used to provide values for certain parameters, unless the user provides details to the contrary; for example, that the date of the flight is ‘today’. Necessary constraints are used to infer less specific information from more specific information, for example, if the arrival airport is known then, in many cases, so too is the arrival city.

Once database access has taken place, the solutions are filtered accordingly to four subintervals:

```
0 ... Min ... Max ... Threshold ...
```

Min and Max describe the optimum range for the number of solutions which can be presented to the user directly. Entries within the interval from Max to Threshold will be tolerated too, but the results summarized. The numbers of solutions below Min or above Threshold are not acceptable for the presentation. One important strategy for dealing with...
with the former case is to use constraint relaxation so that the value of a non-discrete task parameter is relaxed and database access retrieved. For example, the user may ask for information about a flight departing at 10.30 but will accept information about flights leaving just before or after that. Finally, each acceptable solution is presented according to the request template. With (4), solutions to a request for the arrival time will contain the information shown in S3 of (1)². If there are no solutions even after constraint relaxation, then the user is informed that their enquiry has been unsuccessful.

2.4 dialogue techniques

On the basis of semantic and pragmatic functions assigned to user input, an interpreter in the dialogue module applies update rules to the current state of the dialogue model to derive a new state. As a side-effect, task-related information may be passed to the task module, and further goals added. This state is then evaluated to select those goals which provide the locally optimal dialogue continuation.

The dialogue model is composed of goals and contextual variables. These goals are types of dialogue acts which describe intentions of the system in the dialogue. As illustrated in Table 2, some goals are concerned with information transfer, like request and inform, while others, such as close and confirm, are concerned with dialogue control (Bunt 1989). Each dialogue goal is of the form

\[ \text{TYPE} \times \text{CONTEXT} \times \text{STATUS} \times \text{COUNTER} \]

where \text{TYPE} indicates the dialogue act type,

\[ \text{CONTEXT} \text{ the semantic function, } \text{STATUS} \text{ whether the goal is active or pending, and } \text{COUNTER} \text{ the number of times the goal has been realized. The contextual variables indicate the current status of dialogue strategies, such as whether parameters are to be confirmed and the type of confirmation strategy, as well as the status of various contextual parameters, including a 'repair' threshold value for the COUNTER in dialogue goals.} \]

A dialogue state is characterized as a set of instantiated goals and contextual variables. The active dialogue goals are updated on the basis of the semantic and pragmatic functions using rules which are sensitive to contextual variables³. Each type of goal is associated with success and failure conditions as illustrated in Table 3. A goal will

<table>
<thead>
<tr>
<th>Goal</th>
<th>Success</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>confirm</td>
<td>Ø</td>
<td>modified, repeat, reject Ø</td>
</tr>
<tr>
<td>request</td>
<td>new, modified, repeated</td>
<td>Ø</td>
</tr>
<tr>
<td>explain</td>
<td>Ø</td>
<td>Ø</td>
</tr>
</tbody>
</table>

Table 3: Success-Failure Conditions

be satisfied if (a) the semantic input matches the same parameters in its CONTEXT and (b) the assigned semantic and pragmatic functions either do not contradict any of its failure conditions or match one of its success conditions. In (1) the system utterance S2 BA777 from Stockholm? is the realization of two confirm goals — one for the flightid parameter and another for the sourcecity — and both are satisfied by the user's response yes: this utterance is assigned the pragmatic function accept which is not one of its failure conditions. In this way, one function can affect more than one goal. Similarly, a request goal will only be satisfied if the semantic function of the relevant parameter is new, modified or repeated; the goal will fail if, for example, the user simply says (or is interpreted to say) yes. A semantic function which is relevant to a goal but does not match its CONTEXT can also be associated with the goal; this occurs with over-informative responses such as represented in (3). Finally, an explain goal will always be satisfied.

The effect on the dialogue state is determined

²All arrival information is provided and repeated to minimize difficulties which may arise from synthesized speech and reduce the need for the user to ask for additional information.

³A subset of pending goals are updated in a similar manner. This allows pending requests for parameters which the user has provided in over-informative answers to be removed.
using update rules for (a) the satisfaction function of each goal, and (b) the semantic and pragmatic functions now associated with them. Each rule maps from a function to a set of actions to be applied to the goal in the current dialogue state. These rules are sensitive to goal type, the specific semantic parameters as well as the status of contextual variables as illustrated in Table 4. If a goal is satisfied, then it is ‘popped’ from the dialogue state; if it is a close goal, a contextual variable is set indicating that this system should restart after the generation cycle is complete. If a goal is not satisfied and its COUNTER is less than the repair threshold variable, the goal is retained but its COUNTER is incremented; a failed open goal, whose COUNTER is equal to the threshold, results in the system adopting a ‘menu-driven’ interaction style. The first and second rules for the modified semantic function apply when the confirmation strategy is active. In the first, the repair threshold is not met, so a goal confirming the new value is ‘posted’ in the dialogue state, a negative prediction is generated, and the confirmation strategy is set to confirm parameters in separate utterances. In the second rule the repair threshold is met resulting in a spell goal being used to obtain a ‘fresh’ value for the parameter. In the third instance, the confirmation strategy is not active and no action is taken.

It is very straightforward to change dialogue behaviour by changing these rules and contextual variables. For example, if recognition performance was so good that confirmation was unnecessary, then the confirmation strategy could be set to ‘no’. Alternatively, if we decided not to confirm modified information associated with request goals below the repair threshold, then we simply need to add a modified rule specific to this goal type.

In general, the approach allows for confirmation and clarification of user input (to minimize dialogue breakdown), as well as requests for further information (to maximize dialogue progress). Since the application of these rules is sensitive to contextual variables — which are themselves threaded through successive dialogue states — progress can be monitored, and responded to, throughout the dialogue. The effect is that the overall behaviour of the system varies with the degree of success in the dialogue (Eckert and McGlashan 1993; Heisterkamp 1993). If the dialogue is progressing well, the user is permitted considerable freedom; otherwise, the system restricts what the user can say so that the dialogue can recover.

Once the dialogue model has been updated, a subset of the current goals are selected for realization in the next system turn. The selection algorithm uses a classification of goals as initiatives, reactions and evaluations; for example, a request goal is an initiative, inform a reaction, and confirm an evaluation. Depending on the dialogue strategies, different selections may result:

- **evaluation only** select some confirmations depending upon the confirmation strategy
- **initiatives and reactions only** select one initiative and any explanatory reactions
- **evaluations, initiatives and reactions** select an initiative and some confirmations depending on the confirmation strategy, and any explanatory reactions

Goals are then realized in the following order: Reactions (Explanations) > Evaluations > Initiatives > Reactions (Others). This ensures that answers are realized earlier than questions (thus generating adjacency pair sequences) and that explanations precede all other output. Explaining that the system is, for example, repeating a confirmation due to ‘nothing being heard’ makes the behaviour of the system more transparent to the user and subtly influences how they respond. Finally, predictions are calculated for each realized goal. These are used to supply top-down cues.

<table>
<thead>
<tr>
<th>Function</th>
<th>Conditions</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>succeed</td>
<td>type=close</td>
<td>pop, pop, set(restart,yes)</td>
</tr>
</tbody>
</table>
| fail     | rt=less    | rep:1, 
| fail     | rt=eq, type=open | pop, set(menu,yes) |
| modified | rt=less, cs=yes | post:confirm, neg?red, set(sc,yes) |
| modified | rt=eq, cs=yes | post:spell, neg?red |
| modified | cs=no      | () |

Table 4: Dialogue Update Rules (‘rt’ is repair threshold, ‘cs’ is confirmation strategy and ‘sc’ is single confirmation strategy).

---

4Experience taught us that multiple modifications of a parameter are usually an artifact of poor recognition.

5Those selected have the STATUS active; the remainder are pending.
straints on the speech and language components, thus limiting the search space and forestalling recognition errors. Depending on the type of parser and speech recognizer, they can be used to limit or prioritize what can be recognized (by means of dialogue-state dependent n-grams or finite state grammars), or not recognized — predictions can be negative so as to avoid the 'broken record' effect of continually repeating the same recognition error.

3 beyond spoken dialogue management

As spoken dialogue systems for simple information services begin to move into the area of technology, research interest is increasing turning to the integration of spoken dialogue interfaces with other modalities. A **multimodal system** supports interaction with the user through more than one modality, with respect to input and/or output, and with the capacity to interpret and/or generate with respect to the representation of content. Various systems have been developed recently combining speech (or text) interfaces with other modalities (Maybury 1993; CUBRICON, for example, combines speech with a graphics and mouse interface in the domain of mission planning; and Alfresco combines text with hypermedia for art exploration.

The aim of combining a speech interface with a graphical interface is to provide more efficient access to the backend application than the graphical interface alone. Apart from the general advantages of allowing an alternative input and output modality, synergic effects (such as clicking on an object and speaking a command), concurrent tasking for eyes-/hands-busy situations, speech can compensate for some of the apparent limitations of a graphical interface. For example: increased speed of interaction, higher-bandwidth (attention and attitude expressed through stress and prosody, etc), descriptions of objects which are not visually present or are awkward to access in a conventional interface without negation, quantification or temporal expressions, as well as the convenience of reduced descriptions, such as anaphora and ellipsis. Conversely, the graphical interface can compensate for limitations of speech by making immediately visible the effects of actions upon objects, and indicating through the display which objects (and by extension which actions) are currently salient for the system.

Recent empirical studies have suggested that users not only prefer to interact multimodally, but that compared with a speech-only interface a multimodal interface can reduce performance errors, spontaneous disfluencies and task completion time (Oviatt 1996). However, such results must be treated with caution. Firstly, the performance gain is not consistent across all tasks; other studies have shown that task completion time is faster with speech-only interfaces in verbal and numerical tasks. Secondly, these results were obtained using the WOZ technique: response time was always less than 1 second and there were no (simulated) recognition errors. User's perception of, and performance with, speech recognition can be downgraded when faced with the imperfections of present-day technology (Damper and Wood 1995). Thirdly, most studies have focused on speech interfaces for command and control applications rather than more interactive applications where some degree of multimodal dialogue management is required (McGlashan and Axling 1996). Integrating a spoken dialogue interface with a graphical interface for this type of application may introduce new problems; for example, spoken dialogue interfaces are instances of indirect management interfaces (the user delegates some tasks to a software agent) while graphical user interfaces are instances of direct manipulation interfaces (the user is responsible for explicitly initiating and monitoring all tasks). Consequently, it is our belief that we need to build and evaluate such multimodal systems before we can be sure that they match the needs (and abilities) of users as well as telephone-based systems for simple information services.

4 a multimodal system for consumer information

Our dialogue management techniques are being incorporated into a multimodal system combining a spoken dialogue interface with a graphical user interface, which provides consumer information about microwave ovens. Due to limited

---

6 Various definitions of multimodal system have been offered, and are frequently contradictory due to modality being used both to refer to the sensory channel by which information is conveyed (e.g. visual) and the form of the expression (e.g. graphics).

7 This work is part of the OLGA project. The partners are SICS, Stockholm University, KTH, NASA, and Nordvis AB. I am particularly grateful to Yngve Sundblad, Jonas Beskow, Nikolaj Lindberg and Joel Sunnehall for their contributions to this project.
resources, we adopted a 'storyboard' approach to the design of the system (rather than conducting a user study and running WOZ simulations to obtain user requirements) and restricted the first version of the system to allow either speech or direct manipulation input but not both in the same turn. Our design approach resulted in scripts describing how hypothetical users with different needs might interact with the system. A fragment of one script (translated from Swedish), aimed at users with a preference for the speech modality, is shown below.

(5) S1: Whirlpool has five tested microwaves on the market. [Five ovens are shown together with their main properties]
U1: I would like one with a grill, but they are very expensive. Is there anything cheaper with a grill?
S2: Whirlpool has no cheaper ovens with a grill. Here you can see a selection of cheap microwaves with a grill. [A number of ovens are shown, plus a button to 'show more']
U2: Okay, print them out.
S3: You have choose ovens which don't have digital timing. Would you like to known more about digital timing? [Olga holds up a tip flag]
U3: Yes.
S4: [A movie illustrating the advantages of digital timing is shown]. Is there anything else you want to know?
U4: Yes. Show me the Whirlpool ovens again.
S5: [The whirlpools are shown again]

We also developed a simple mockup of the user interface in order to get a clear idea of what the system would look and sound like for the user. A snap-shot, corresponding to S2 in (5), is shown in Figure 1. In addition to speech and language components, the system is composed of three other components: a direct manipulation interface which provides graphical information and widgets for navigation; an animated talking agent whose speech is synchronized with its lip movements, and who performs gestures; and a dialogue manager for coordinating interpretation and generation in both modalities. The dialogue manager is the same dialogue manager described in Section 2 but augmented in four fundamental ways.

4.1 non-speech input and output

In order to manage multimodal dialogues, input and output need to be informationally compatible at the dialogue management level. A user may provide input via buttons in the interface and the system generate a spoken response; or a user may reference in speech an object which the system has realized graphically. Consequently, all input and output is represented in the semantic description language discussed in Section 2.2. Rather than ask the system to print out the products on display in U2, the user can have pressed the 'show more' button. This button, part of the graphical realization of the system goals, is directly associated with a show action and a semantic description of the other microwaves. Using this technique, interpretation of graphical input is simpler than interpreting speech input because there can be no recognition errors and, since a specific semantic description is already provided, no difficulties in determining what the user is referring to.

4.2 referencing visual objects

Like simple information service dialogues, user actions can be interpreted as providing parameters for the information service as with U1 in (5), or as dialogue controls like U3. Consumer service dialogues also introduce a set of actions used to execute commands on objects and navigate around the information space. For example in U2, the user asks the system to execute a print command on visually-present objects and in U4 the user asks the system to show previously dis-
played objects. Such actions are not observed in simple information services since the service is completed once the information has been presented to the user.

In order to characterize the referential descriptions in these utterances, we use the existing modus feature of our semantic representation; for example, *them* will have the features pro:pro, def:ndef, number:pl indicating that it is a definite plural pronoun, and the whirlpools the features pro:nonpro, def:ndef, number:pl together with a manufacturer:whirlpool parameter. Reference resolution is still based on finding compatible, accessible concepts in the discourse model for definite expressions, but three new semantic functions are required:

ref.success appropriate antecedents have been found; this may be a single object or a set of objects which form part or all of an existing semantic description

ref.ambig more than one appropriate antecedent has been found

ref.failure no appropriate antecedent has been found

At the dialogue level, the first function allows the command to be executed immediately, while the second and third result in clarification goals being added to the current dialogue state. Interpretation of U2 and U4 in (5) results in ref.success assignments: *them* refers to objects (from the preceding system turn) which are currently visible; and the whirlpools to objects (which are no longer visible) from a system turn earlier in the dialogue. While this simple technique has been successful so far, we realize that a more sophisticated approach may be necessary; for instance, an approach where the accessibility of an object is a function of the priority and temporal persistence of the modality introducing it (McGlashan and Axling 1996).

4.3 modality selection

The dialogue state is updated and goals selected for realization according to the principles discussed in Section 2.4. Extensions have been required to select between the three realization modalities; apart from speech and graphics, the Olga animated agent can perform a limited range of gestures such as pointing at the graphical display, looking at the user, and facial movements such as eyebrow-raising, smiling, looking sad, and so on. In general, modality selection is defined in terms of characteristics of the output information, and the expressiveness and efficiency of the alternative modalities for realizing it. Given the absence of many theoretically-motivated selection principles, most multimodal systems employ domain-appropriate heuristics; for example, the AIMIT system uses rules based on characteristics of the user query. Similarly, we currently use rules based on the type and content of the system goal.

Goals with a control or feedback function are realized in speech and gesture: for example, success in understanding user input is indicated with a head nodding gesture, while failure is indicated by speaking an explanation of the failure together with raised eyebrows and the mouth turned down. Product information is presented in speech and graphics; detailed product information is displayed while the agent provides a spoken overview as illustrated in Figure 1. Finally, since the content and structure of the graphical display can have a significant effect on user utterances — an unstructured display can easily result in speech disfluencies and hence speech recognition problems — the realization of inform goals is followed by the agent looking at the user. In situations (like S4) where the system presents a video explanation of microwave features, the visual realization is followed by a spoken realization of a 'dialogue continuation' goal designed to direct the user's attention back to the interaction with the system.

4.4 navigation and filtering

At the task level, the system uses the techniques described in Section 2.3 to update its task model and filter database solutions. When access results in too many solutions to present on screen, they are filtered according to task-specific strategies such as whether the product has been included in consumer tests. When there are too few solutions, constraint relaxation is used to offer information about alternative products; in S2, the system realizes a goal explaining that the manufacturer parameter has been relaxed in order to satisfy the user's current request parameters. This has required an extension to deal with relaxa-

---

8With indefinite expressions, such as *show me a whirlpool oven*, a new object is created in the discourse model, and a database lookup executed.
tion of discrete parameters. Another extension is the addition of functionality to allow the system to provide an explanation of desirable product features. This behaviour is triggered when the user has selected products for printing. For example, the user has selected a product without digital timing, then the agent holds up a 'tip' flag and offers an explanation of the feature as in S3.

5 Conclusion

Effective dialogue management can compensate for limitations of speech and language processing by providing a cooperative interaction with the user. We have described techniques of spoken dialogue management for simple information services. The layering of semantic and pragmatic interpretation, combined with flexible dialogue rules, provide fine-grained control over the system's behaviour. This approach has been tested and evaluated in a number of speech-only systems with remarkably few changes.

The approach has now been incorporated in a multimodal system combining speech and graphical interfaces. Interpretation and generation of graphical input is based on the same semantic and pragmatic functions required for spoken language. Since the application domain includes navigation and command utterances not observed in the simple service applications, extensions have been made to the reference resolution algorithms. Additions have also been required for determining in which modalities system goals should be realized. However, there are a considerable number of multimodal issues which we have yet to address even within the (relatively) simple domain of consumer information services. The majority concerned our ignorance of what functionality user actually need, and how they will really react to this type of multimodal system. User trials will allow us to begin to address these issues.

References


---

This feature is not a design decision on our part, but a requirement from our sponsors.
A CONVERSATIONAL AGENT TO NAVIGATE IN VIRTUAL WORLDS

Pierre Njugues, Christophe Godéreaux, Pierre-Olivier El Guedj, & Frédéric Revolta
Institut des Sciences de la Matière et du Rayonnement
GREYC
6, boulevard du Maréchal Juin
F-14050 Caen, France
eMail: pnugues@greyc.ismra.fr

ABSTRACT

This paper describes the prototype of a spoken conversational agent embedded within a virtual reality environment. This prototype – Ulysse – accepts utterances from a user enabling him or her to navigate into relatively complex virtual worlds. The paper first describes what we can expect from such an interface in the communication quality between a user and virtual worlds. Then it describes Ulysse’s architecture which includes a speech recognition device together with a speech synthesizer. Ulysse consists in a chart parser for spoken words; a semantic analyzer; a reference resolution system; a geometric reasoner, and a dialogue manager. Ulysse has been integrated in a virtual reality environment and demonstrated.

INTRODUCTION

User interaction in virtual environments has almost always been undertaken with more or less sophisticated pointing devices. These devices enable to move in horizontal and vertical planes and to rotate. They also enable to point at a specific object and to “teleport” the user to it. Finally, they enable to interact with objects of the virtual world: to move them, trigger them, etc.

Navigating in virtual worlds – virtual reality – with devices such as mice, space balls, is one of the trickiest issues for new users. Certain motions are difficult and a novice user can easily get seasick with her/his “body” upside-down within a two minutes session. However, in many situations, pointing devices enable a fast and accurate interaction.

Speech interfaces are beginning to appear in virtual or simulation environments to ease interaction (Karligren 1995; Bolt 1980; Ball 1995; Everett 1995). Spoken interaction in a virtual environment requires to complement conventional pointing devices, to coordinate both means of interaction and to leave the user the choice – the initiative – of interacting means she/he wants to use. While it does not seem desirable to try to substitute completely these devices – it is sometimes much easier to point at an object than to describe it in a verbal way – there are some situations where we prefer “to say it” rather than to “do it”.

Human-machine dialogue requires several relatively generic linguistic modules or devices such as speech recognition systems and speech synthesizers, syntactic parsers, semantic analyzers, and dialogue managers (Allen 1994a). In a virtual environment, speech is only one mode of interaction – possibly a minor one – and some adaptations must be made to classical dialogue architectures. Pointing devices must be smoothly integrated with speech. This notably implies means to resolve deictic references that is coordinated with pointing devices and hence to reason about the geometry of the scene. Beside, the architecture must be complemented by an action manager that will make the user feel comfortable with her/his “body” motion in the virtual world.

COOPERATIVE WORK,
TELECONFERENCING AND,
VIRTUAL REALITY

Computer Supported Cooperative Work (CSCW) gave the framework of the Ulysse project
and was part of the European commission COST-14 project (CoTech 1995). CSCW research attempts to determine how a computer can help people better work together on a project, to design a product, to take a decision, etc. across a network. CSCW tools enable notably to share documents with multi-user editing tools, to discuss design strategies using shared white boards, to communicate using real time teleconferencing: text, audio, or video.

Text teleconferencing is now widespread on the Internet and notably consists of forums you can connect to and participate in to work, discuss ideas, make friends, etc. Video teleconferencing that broadcasts participants’ image is certainly an improvement over text and audio provided there are only two parties. When the number of users increase, the screen gets cluttered with the faces of the different parties and the communication tends to be difficult: who is talking to whom?

Spatial metaphors have been identified as a mean to improve the comprehension of teleconferencing and Internet forums. It resulted into the re-creation of meeting rooms or more complex scenes using virtual environments. Such environments enable users to meet in a virtual room, to move about or get from one room to another. There, participants are embodied within these virtual worlds using more or less realistic 3-D icons called embodiment, representation, or avatars (Benford 1995). As a result, users can immediately realize the complexity of a situation. The counterpart is that it is much more difficult to interact with the interface.

THE TASK

We investigated spoken interaction in a virtual environment using the Distributed Interactive Virtual Environment (DIVE) (Andersson 1994)

![Figure 1 A Snapshot of the Ithaque World](image)

from the Swedish Institute of Computer Science (SICS). DIVE enables to build virtual worlds where users can connect from a remote location, move into, and meet other participants. Participants share the same geometric model of the world with a different point of view. Modifications of the world from user interactions are replicated to the other participant sites to keep the world consistent.

We collected a corpus of dialogues involving two experienced and two novice users. We recorded these dialogues from four interaction sessions (Godéreaux 1994) in a world – Ithaque – similar to that on Fig. 1. Each dialogue involved two participants: the interacting user and another who played the role of the agent by acting on the virtual world. We plotted two scenarios. In the first one, novice users had to move and discover the world and in the second one, more experimented users had to discover a treasure hidden in the world. Users were supposed to be alone in the world – with no other connected participants.

In comparing mouse and speech interactions, we found that mouse navigation was a major difficulty. In subsequent sessions, we even realized that most novice users were unable to go around the house. More precisely, if navigation can be relatively easy
in a given plane, it is much more difficult to align or to carry out a circular motion. In addition, it is impossible to look at a specific location while moving using a single mouse. This makes some motions clumsy, for instance when the user is going round an object.

In contrast, many motions are easy to formulate verbally (Table 1; Table 2) and coordination of voice and mouse input enables a user to roam all the recesses of the virtual world. In summary, we found that dialogue interfaces can improve the usability of virtual environments. They ease navigation and bring a new channel of interaction.

| A    | nous sommes connectés au monde robot. |
| U8   | tourne sur toi-même.                 |
| A    | vers la droite ou vers la gauche?    |
| U9   | vers la droite.                      |
| A    | voilà.                              |
| U10  | prends de la hauteur.               |
| U11  | arrête de monter.                    |
| U12  | monte.                              |
| A    | oui.                                |
| U13  | stop.                               |

Table 1 Dialogue Excerpt

| U45  | va jusque là.                        |
| A    | je me dirige vers la montagne.       |
| U46  | fait le tour de la montagne.         |
| A    | oui.                                |
| U47  | retourne sur l'île précédente       |
| A    | je ne connais pas l'île précédente.  |
| U48  | regarde à droite.                    |
| A    | voilà.                              |
| U49  | encore.                             |
| U50  | c'est ici.                           |
| A    | je me dirige vers la montagne.       |

Table 2 Dialogue Excerpt

**ULYSSE’S ARCHITECTURE**

Ulysse takes the form of a conversational agent that is incorporated within the user’s embodiment. Ulysse’s overall structure is similar to that of many other interactive dialogue systems (Allen 1994b). It is inspired by a prototype we implemented before (Nugues 1993; Nugues 1994) and features speech recognition and speech synthesis devices, a syntactic parser, semantic and dialogue modules. Ulysse’s architecture is also determined by the domain reasoner and the action manager. At the difference of TRAINS (Allen 1994b, p. 18), deindexing is closely tied to dialogue and to reasoning capabilities.

Ulysse’s capabilities are relatively specific and concern only navigation. Ulysse assists the user within the world by responding positively to motion commands. Ulysse acts consequently and transports the user within the virtual environment on her/his behalf. In other projects such as (Karlgren 1995) and (Everett 1995), more general capabilities are implemented that allow the user to talk to the “world”. The corresponding agents act upon the context and usually navigate (move the user embodiment), manipulate virtual objects, or answer to queries.

Understanding navigation commands requires to resolve the many deictic references that occur in the conversation and to reason about the geometry of the world. Ulysse’s architecture is complemented by a reference resolver that works in coordination with the user's gestures enabling her/him to name and point at objects and a geometric reasoner to understand the world. The navigation is completed by an action manager that brings the user in a relatively continuous motion where she/he wants to go.

**SYNTACTIC PARSING**

Speech is recognized using the IBM’s VoiceType commercial device. VoiceType is operating on isolated words – the speaker must pause between words – and is primarily intended for report dictation. We have chosen this device because it can process French and can recognize several other European languages with a vocabulary of up to 30,000 words. A chart parser is connected to the recognition device output and takes up the words. This chart (El Guedj 1994) adopts a classical bottom-up algorithm with a dual syntactic formalism: It can operate using phrase-structure rules and a dependency formalism (Tesnière 1957).
A constituent grammar was used to encode the lexicon - 350 distinct words - and phrase-structure rules accepting all the 400 utterances of the corpus (Godédeaux 1994; Godédeaux 1996). The lexicon is using parts-of-speech that are a variation of Multext categories (Véronis 1995). We retained as features only those that were relevant for French.

Phrase-structure rules are rewriting the utterance structure using unification constraints and non terminal categories such as noun groups, verb groups, prepositional groups, determiner groups, adverb groups, adjective groups, etc. Rules were adapted to accept missing and unknown words. They include a large number of prepositional, adverbial, and demonstrative locations that are ubiquitous in spoken language.

Utterances correspond to four main clause types: orders, questions, statements, and subordinate clauses, and also to phrases without a verb. The user segments her/his utterances using a "push-to-talk" scheme and signals the end of them by pressing a button. The analysis results in parse trees - up to eight in our corpus -. They reflect the syntactic or semantic ambiguity of the utterance.

**SPEECH ACTS AND SEMANTICS PROCESSING**

Utterance parse trees are first mapped to speech acts representing mainly navigation commands, such as *va dans la maison*. Other conversation acts that are identified by Ulysse are:

- deictic clarifications, such as *celle-ci*
- motion modifications, such as *plus vite*
- motion repetitions, such as *encore*

Semantic interpretation considers only navigation commands. It splits the utterance into clauses, and tags constituents from the chart parse tree with syntactic functions. Functions correspond to classical subject, object, or adjunct that are sub-classified using ontological categories. This stage also attaches modifying adverbs to their head words: verbs or other adverbs. Semantic annotation of verbs is related to the motion - the navigation - that is desired by the user and to space description. Considering our corpus and lexical sources (Bescherelle 1980), we divided them into five main navigation categories:

1. **go** (*aller, avancer, entrer, monter, sortir, etc.*) corresponds to a change of location with a possible rotation of the embodiment;
2. **return** (*revenir, retourner, etc.*) in this category, the object visibility does not matter;
3. **rotate** (*se tourner, regarder, pivoter, etc.*) corresponds to the rotation of the user's embodiment head;
4. **stop** (*arrêter, stopper, etc.*)
5. **continue** (*continuer*)

Assigning these semantic tags is sometimes ambiguous. Compare: *retourne-toi* that belongs to
the 3rd category and retourne dans la maison that belongs to the 2nd. We carried out disambiguation using verb syntactic sub-categories: i.e. transitive, intransitive, or pronominal, that we encoded as unification constraints.

As a result of this stage, each sentence is transformed in a list with as many items as there are clauses. Each clause is mapped to a structure whose members are the subject, verb group, and a list of complements. Each complement being annotated with a semantic tag: time, manner, location, etc. Verbs groups are also annotated with a motion tag and packed with possible adverbs and clitic pronouns. Ulysses maps to the same command wording differences such as:

Avance (go on)

Je veux avancer (I want to go on)

Peux-tu avancer? (Can you go on?)

Je veux que tu avances (I want you to go on)

When utterances consist into several clauses, they are concatenated and possibly rearranged according to “connectors”. These connectors are associated with list operators such as append, delete, replace, or insert. Connectors can be adverbs, conjunctions, or syntactic forms. For instance negation adverb not in the sentence Monte non descends results into the replacement of first verb. Adverb puis in the sentence: Monte sur la maison puis va devant l’ordinateur results into the appending of the second action. Gerund en passant in the sentence: Va vers la maison en passant devant le drapeau results into the insertion of the last motion before the first one.

The logical form list is post-processed to relate it to a sequence of basic actions. According to the verb type, a clause can be expanded in one or several basic actions (up to three). For example:

monte sur le drapeau, corresponds to

1. go onto OBJECT (flag)

retourne dans la maison, corresponds to

1. turn back
2. go into OBJECT (house)

GEOMETRIC REASONING AND REFERENCE RESOLUTION

The reference resolution module de-indexes the sequence of action predicates resulting from the semantic interpretation. Object references are ubiquitous in the corpus and in all the subsequent experiments we conducted. It includes specific parts, such as: va devant l’ordinateur, entre en la voiture à gauche de la maison, plurals: dirige-toi vers les cubes, multiple choices: va dans la maison – with several houses –, and deictic sentences such as: va ici. References must take into consideration the state of the world database, the user’s position in the world, together with the interaction history.

Associating a name to an object is sometimes tricky. Users can have a different wording to designate the same thing. Geometric databases may also consider certain objects as compounds or hierarchy although they form unique entities in the user’s mind. For instance a house can be represented as a single entity, as windows, doors, walls, etc. or as a set of polygonal lines. In addition, it is important to differentiate objects that have a front and a back from other non oriented objects.

In the present prototype, we addressed the naming problem by carefully associating a name with the entities of the world database. We structured the database to keep the most consistent relations between names and world entities according to our corpus. We also gave a main orientation to each object if it could have one and references axes originating at its gravity center. The overall shape and gravity center of objects enable to compute a kind of acceptable distance to position a user relatively to an object when she/he wants to move to it: close for a small object, farther for a bigger one. Gravity centers enable also to approximate a group of objects to a unique entity.

Ulysses references objects by constructing a list of compatible entities from the geometric database when an object name occurs in an utterance. When several objects are candidate, it resolves the ambiguity using a salience algorithm similar to that of Huls (1995). Two criteria are taken into account according to the verb type:

- Visibility of objects from the user point of view;
- Focus coefficients that reflect object interaction histories.

Visibility of an object results from the intersection of the user’s visibility cone with the world database. If there are still several objects that remain candidate, Ulysses considers the Karigren’s focus (1995) that is attached to each object and retain the greatest. The focus of an object is incremented each time the user interacts with it –
mentions it or points at it - to become the greatest of all the foi. Although apparently simple, this resolution scheme yields accurate results.

**DIALOGUE PROCESSING**

The dialogue module monitors the turn taking and the sequencing of Ulysses modules. It corresponds to getting the utterances, processing them, and executing them. The dialogue module manages the syntactic ambiguities by sequentially providing the semantic interpreter with the parse trees until it finds a correct one. It then passes the clause list to the reference resolution manager.

If the references can be resolved, and there is only one solution, the list of actions is passed to the action manager. If there are several possibilities, the situation is clarified by the dialogue manager and notified to the user using a spoken message. We implemented a simple scheme to handle multimodal clarifications using the focus coefficient. The dialogue manager asks for a pointing designation and the referencing process is repeated with the last semantic interpretation. The pointed object is designated without ambiguity since its focus is the highest of the list. If there is no referencing solution and if there are other syntactic parse trees, the dialogue manager gets another semantic interpretation and passes it to the referencing module until the parse tree list is empty. The user is then signaled of the failure.

If the action can be completed (corresponds to implemented navigation commands), the action manager will go on and move the user's embodiment. Otherwise the dialogue manager restarts the process with the next parse tree. An example of it is the sentence *Prends de la hauteur* (Gain height) that can be a location and also be interpreted as a transitive verb and a direct object (Take height) that cannot be executed.

If an utterance corresponds to executable commands, the system will acknowledge them using a random positive message while carrying them out. Otherwise, once all the possibilities are exhausted, the dialogue manager rejects the utterance, indicating the cause. The natural language generator uses template messages and possibly selects a random one. The parser is also used by the generator to check the correctness of the agent answer.

**THE ACTION MANAGER**

The Action manager queries the Geometric reasoner to convert the referenced list of actions into a sequence of position coordinates. The reasoning is based on the verb category of each item of the action list. These verbs correspond roughly to the categories *Change of location* and *Change of posture* described by (Sablayrolles 1995). According to the category, different kinds of actions are undertaken:

- go corresponds to a change of location and to a sequence of space positions.
- turn corresponds to a rotation of the whole body or of the head.
- return corresponds to a turn and to a change of location.
- stop will stop the action
- continue will resume the action

Computation of space positions takes into account the triplets verb, preposition, and object shape. It enables a user to go to a house and to stop while this house is still largely visible and go to a chair in a similar way but to get in fact closer.

In addition, head and body are articulated and rotated separately. This enables the user to move in a direction while looking in a different one. When going around a house, a simple motion could implement a four corner motion with a reasonable distance, but the user would lose the eye from the object he/she probably wants to consider. In our prototype, the sight is directed on the main object of the utterance, for example when going around a house, the user will keep an eye it. This makes the user feel more comfortable with her/his embodiment.

The action are implemented by dividing the action into small sub-motions using a callback function. The callback adjusts the length of the sub-motion and enables to vary the speed.

**A DIALOGUE EXAMPLE**

The action manager enables exchanges such as the sequence on Fig. 3.
User and agent utterances

Bonjour Fred,
bienvenue dans le monde
"Ithaque"

D'accord

Retourne toi

Va vers les deux voitures à gauche de la maison

Voilà

Voilà

Regarde la maison

Tourne à droite
Oui

Voilà

Va devant cette voiture

Voilà

Il y en a plusieurs

Tourne à droite

Va vers les petits cubes
Voilà

Va derrière

Retourne devant la maison

Voilà

Encore

Va derrière
CONCLUSION AND PERSPECTIVES

We have presented a conversational agent – *Ulysse* – that enables a user to carry out relatively complex motions within a virtual environment using voice. Our prototype consists in a commercial speech recognition device, together with a speech synthesis circuit. It relies on a modular architecture embedded within a virtual environment. The entities of the prototype are to process syntax and semantics, together with dialogue and actions that are resulting from them. We demonstrated the prototype to various people, notably at *La Science*
en Fête, the national science open day in France where it gathered the enthusiasm of a young attendance.

This project has been developed within the framework of the COST-14 program on CSCW tools from the European commission. We think this type of conversational agent has other application perspectives. Virtual reality environments are blooming – they are now included in many Web browsers – and spoken interfaces could complement and sometimes substitute conventional pointing devices.

At Caen, we plan to adapt our agent to the virtual reconstruction of the Ancient Rome that is being undertaken from a plaster model. We are also adapting the agent to the spoken manipulation of brains reconstructed from MRI images. In conclusion, we think that this kind of agent offers perspectives to experiment tools and theories in dialogue and space linguistics.

REFERENCES


BESCHERELLE, L'art de conjuguer, Hatier, 1980.


COTECH: Minutes of the COTECH Workgroup: Virtual and Augmented Environments for CSCW, Department of Computer Science, University of Nottingham, Nottingham, England, 1995.


WHICH PROCESSES TO MANAGE HUMAN-MACHINE DIALOG?

Anne Vilnat
Groupe Langage et Cognition
LIMSI - CNRS
B.P. 133, 91403 ORSAY Cedex, FRANCE
Phone: 33 1 69 85 80 03
Fax: 33 1 69 85 81 33
E-mail: vilnat@limsi.fr

Abstract:

Which processes are necessary to interpret an utterance inside a dialogue, and which are those that allow to produce another utterance that is relevant inside the context of this dialogue? It is the question I will try to answer. The different works concerning dialogue systems indicate how various are the processes which may be necessary to elaborate an answer.

After examining the different approaches which have been developed, I will present the interpretation component of the system we develop. It is based on the fact that the meaning of an utterance in a dialogue proceeds from analyses on different points of view, namely the informational, the conversational and the intentional context. I argue that these analyses must be separated, because they are based on different knowledge sources, whereas the meaning is obtained from a combination of the results of these analyses. Before considering the intention recognition, it is necessary to establish the informational and conversational coherence, in order to reduce the complexity of this recognition. I will present how these two kinds of coherence can be handled. Therefore, the topic and communicative intention processes are described, with the knowledge they require. Their collaboration, through a dialogue manager, is exhibited with its results in context-changing actions.

I An overview of dialog systems

The first developed dialog systems were supposed to be able to directly answer the question asked by the user. In this context, it was quite impossible to manage a dialog more complex than a simple exchange. The user must then be able to give the complete information necessary to find the answer. But this limitation was too strong, even to realize simple tasks. Another possibility was to develop a dialog inside a micro-world such as in SCHRDLU (Winograd 75).

Then, following GUS (Bobrow, et al. 1977), dialog systems have been developed whose goal was to give information in a precise domain. It was possible to manage some exchanges. But a strong hypothesis remain : the dialog must concern the only topics directly related to the task, that means those which are modelised inside the frame data structure. Even in dialogs which are often considered as simple, such as train information, it is difficult to predict all what the user will say. And, hypothesising that what is not predicted is not useful, may lead to dialogs which are not very cooperative. And its seems rather difficult to increase the system abilities by adding other frames which will potentially be filled during the dialog.

Quite at the same time have appeared systems based upon plans, following (Allen et Perrault 1980). The notion of speech acts and the fact that dialogs are implicit by nature are enlightened.

Before presenting more recent systems based on the same ideas, let us examine the different kinds of dialogues which are studied, because that implies different approaches of this problem, related to the question of the evaluation of these systems.

1) What kinds of dialogues?

Inside the dialog community, it is now possible to distinguish two approaches : (a) the first one consists in determining the capacities the system must have from an analysis of the application to realise, often based upon "wizard of
Oz" corpus\(^1\); (b) the other one is to examine "interesting" phenomena and try to decide how to take them into account, even if dialogs in which those phenomena are illustrated have not been really encountered in a corpus.

The limitations imposed by these approaches, and their consequences on the design of the systems are developed below. To conclude on this point we will examine if it is pertinent to establish some differences between dialogue systems, depending on the kinds of applications in which they are used.

1.1 Corpus

The asks for new or existing corpus are more and more frequent. Without refuting the interest to obtain this kind of tools, it is necessary to be conscious of their limitations to correctly use them. A corpus should be used to produce some guidelines, before developing an implementation or to verify some hypotheses, if the corpus has been collected with this goal. What we do not agree with is the more and more frequent attitude which consists in funding or completely validating a study only via what is observed in the corpus, which are often collected with some inherent defaults, specially when they are "wizard of Oz" ones. Then, the first default concerns the person in charge of the machine simulation: the limitations and problems he detects are generally not the same than the ones encountered by the real system: he cannot find the information as quickly as the system is able to do, but it is very difficult for him "not to understand" some user's asks, and to continue the dialog, as if he has not understood. Another default concerns the user who is supposed to use a real system. In some cases he only has a scenario to follow, and not a problem he really wants to solve. This greatly modifies his attitude, both in the way he asks his questions and his reactions to the answers he obtains. Even when the experimentation is more realistic, that means when the users are "real" ones, the tests are not long enough to be faced with users who will not limit their requests, because they are used to use it.

Consequently, we do not agree the attitude which consists in refuting the utility of trying to take into account only because it has not been observed in a corpus. We think that it is impossible to ask a user to limit himself to a part of the language, which we can never simply define.

It is obvious that we must also prevent to fall in the opposite extremity: it must be possible to manage a simple problem without requesting too sophisticated (and therefore which may need too much time) tools. This may be solved by the use of adapted software architectures.

1.2 Evaluation

An important problem currently arises inside the community of researchers working on dialog systems: how to evaluate dialog systems? We can notice that it is to try to give an answer to this question that a lot of works are currently directed to corpus collects, so as to validate the dialog systems on their ability to manage such dialogs.

The ability to evaluate our work would really be interesting, but the way to do it does not seems to be so evident. If it is easy to understand how it has been done for speech recognition, it is very difficult to determine how to do it for dialogs. It exists evident criteria to decide if a speech signal has been correctly decoded: the results may be compared with the words which have been uttered. But how to decide if what has been said during a dialog has been correctly interpreted: then you will have to compare the results obtained by different interpretation modules. These results are internal representations more or less detailed, based on different formalisms. So it is not easy to compare these results, the only solution being to take into account their adequation to the forthcoming processes which will use them. It is impossible to evaluate a representation in isolation of the process which will use it. Then, a solution would be to globally compare the interpretation and the following processes, which uses it. That means to compare the different systems on the basis of the reaction they produce in a given situation. Thus the evaluation problem leads to the determination of the adequate criteria allowing to discriminate what is a good answer, or what is a good dialog. This question is a priori simple but cannot obtain a simple or general answer. Which criteria do we want to take into account, or in other words, an answer is better:

(a) if it is the more complete one, but then it must still be clear,
(b) if it is the more concise, but it must still be understandable,
(c) if it is the one which is nearest to the final solution, but how to determine that during the dialog,
(d) if it is the one that request the minimum response time, but then what is its relevance...

This list, which is probably not exhaustive, indicates how it is difficult to solve this problem. The definition of a good dialog system which seems to be the most adapted, would be to say that it is the one that a user would be pleased to use... a second time! Unfortunately this definition, which would allow an *a posteriori* selection, is not very useful to design a system.

Thus, without denying the interest of the works aiming to define those criteria (particularly those developed for ARPA\(^2\)), it seems very useful to work on the definition of the necessary components (processes to implement) of a dialog system which is not tailored to a given application, and which is perhaps not the best to solve some simple cases.

To conclude, to evaluate a dialog system, and to compare its efficiency with others currently seems to be a task which is quite impossible, and perhaps not yet useful. We do not want to say that this problem must definitely be ignored, but that it is not the prior one to examine considering the current state of the art in the domain.

1.3 Is a dialogue typology useful?

Ignoring informal dialogues in which the machine does not appear as a wished interlocutor, the dialogues we are interested in may be classified in two classes: information seeking dialogues, and dialogues in which the interlocutors collaborate to execute a task.

In the first case, the user has a particular problem to solve, and to do so he requests information, which will allow him to correctly execute the *task*\(^3\) he is trying to achieve. It is the framework in which are designed the information seeking, or retrieval systems, as well as the ones in which the system play the role of councillor.

In the second case, both the user and the system cooperate to the realisation of the task. It is obvious that the state of the task will be in constant evolution during the dialogue, and that these modifications will play an important role in the dialogue. Command dialogues, or even some teaching systems belong to this category.

The differences perhaps seems a priori important, but a detailed examination shows that it is not the case. For example, to mention only this point, which seems the most different, the notion of a task to execute is always present, and will request a precise representation, even if this task will be executed after the dialogue. The knowledge to represent as well as the reasoning to develop are sufficiently similar to necessitate the use of the same set of processes.

The last point on which I want to insist concerns the fact that it does not exist dialogues which are simpler than the others, only because the domain on which they apply is more limited. Actually, the current limitations of the systems we are able to develop imply that some problems are not taken into account. But as soon as you give the user the possibility to use his own language, the whole complexity of this language may potentially appear: you cannot a priori restrict someone to only part of his language.

Let us now come back to the plan based systems.

2) The planning approach

The more recent researches developed inside this paradigm show that different kinds of information are taken into account: (a) the knowledge directly related to the action which is the subject of the dialog, (b) the knowledge concerning the evolution of the mental states of the interlocutors (their beliefs and intentions), (c) the knowledge concerning the fact that the interlocutors are engaged in a dialogue, and this has implications on the texts constituting their utterances.

As shown in (Pollack 1990), planning in a dialogue is a process related to the interlocutors mental states, which necessitates other information than common knowledge to execute actions, which are often called *recipes*. It is not because you know a recipe to kill a president (by shooting him) that you will use it to recognise the user's plan: you need information on him, and particularly on his mental state, to suppose that he will use such a recipe. She shows that to infer an invalid plan, you must assume that the user's beliefs are different from yours: so the system must be able to manage those different knowledge sets.

The works developed by Grosz, particularly with Kraus (Grosz et Kraus 1993), are based on the same concepts. Inside the plans, the distinction is clearly established between what concerns the mental states of the one who is elaborating the plan (which really constitutes the plan) and what is related to the realisation of the *recipe* to execute an action. In this
system, the different kinds of knowledge are identified, but are fully encapsulated, and thus the reasoning process becomes very complex.

(Litman et Allen 1990) establish a distinction between the discourse intentions and the common sense plans, which are the objects of the intentions. In turn the common sense plans are classified in domain plans (to modelise the task) and discourse plans (which are consequences of the execution of a domain plan or concern discussions about this domain plan). Thus the knowledge related to the domain is separated of the knowledge concerning dialogue itself. But they are managed inside the same reasoning mechanism.

With rather similar intentions, (Airenti, et al. 1993) show that you need to distinguish two kinds of co-operation in dialogues : co-operation based on conversational rules, and co-operation based on behaviour rules. Distinguishing between conversational games and task related games, allow them to manage dialogue phenomena. To maintain a conversation, you need only to follow conversational rules, but to cooperate, the interlocutors must act in agreement with plans at least partially shared. The conversational game encapsulates the behaviour game, and acts as a set of meta-rules via the task resolution.

(Traum et Allen 1994) propose the notion of discourse obligations, which give a priority to some actions, when some obligation sources have been recognised. In these obligation rules are treated the knowledge directly related to the dialogue. The planning will take place after the examination of potential obligations. The distinction allow them to take into account more dialogue phenomena, without complicating the planning system.

In these different systems (except in the last one), we see that the planning is the only reasoning process, that must take into account the different aspects of dialogues. Thus this process, which by essence is complex, becomes more and more difficult to implement.

Another point to underline is the fact that the work being centred on this process, they do not more give attention to the whole process. In particular, the entries of the system is often reduced to the representation of an action potentially obtained by a parser. They forget that the linguistic information of an utterance may give more precise information than a simple action.

After this discussion concerning the different approaches concerning dialog managing, the following part will illustrate our decision to separate the treatment of the different kinds of knowledge in different processes, piloted by a dialog manager. Our motivation is that the knowledge and the reasoning on this knowledge is different enough to be simplified by separate processing. The work of the dialog manager in this framework is to decide which information must be taken into account in priority. Only the interpretation part of this system will be presented here because it the one which is more developed one.

II Our dialogue system

We have defined a dialogue system decomposed in several modules, some of them dedicated to goal computation and planning and others handling different aspects of context. A dialogue manager makes them collaborate and decides which reaction is the most pertinent after consultation of these modules. Our work takes place in the paradigm of task-oriented dialogue, where a user seeks information stocked in a database.

To be flexible and co-operative, a dialogue needs an analysis from several perspectives to capture the meaning of an utterance in the context. We will show that this meaning depends on the syntactic form, the informational content and the place of the utterance in the dialogue. Thus, our analysis is not only realised in a planning perspective, i.e. does the user communicate a goal?, does he achieve part of a current plan?, but also in terms of communicative implications. This approach allows us to maintain a more co-operative dialogue. Therefore, an utterance is interpreted from three points of view : an informational, a communicative and an intentional level. The informational analysis is dedicated to the informational coherence by means of a thematic interpretation. The communicative one represents the level of communicative obligations, captures and maintains the structural coherence of the dialogue by means of a communicative intention analysis and an interaction management. The intentional one classically corresponds to goal computation and planning activities.
intervention forward, here 'a French department' is evaluated as belonging to the same topic than S2b. The interpretation proceeds from the conversational and the informational coherence. The first aspect is managed by the communicative intention analysis and the second by the thematic interpretation (we will see later how they work). With this analysis, co-operativeness is maintained because the system can resume the conversation concerning the main goal of the user, i.e. to speak to Mr Smith, and give a complete answer. We can notice that the system acts according to the conversational principles (Grice 1968a). This results in the first two parts of the answer: 'yes' is a formal reaction to the linguistic form of the user's intervention (acting as a cohesion mark), and 'Paul Smith from the French department' makes explicit the inference made by the system, otherwise the answer would be incomplete in the sense of Grice's maxims.

The meaning of an utterance results from a pragmatic analysis, and linguistic clues are used but never preferred to contextual ones. Otherwise, this would lead to interpret U2 (fig. 2) as a yes-no question where the user is supposed to develop a new plan to find an answer to S1b. The system would react to this new sub-goal and its only possible answer to the last user's question would be yes, there is. We argue that a more complete answer, such as 'yes, and there is a Mr Paul Smith belonging to this department', is impossible without interpreting the user's utterance as a possible answer during the reasoning. By what other means, would a system be able to form the database query leading to this answer? From the system’s point of view, the user’s intention is to obtain a piece of information that will help him to find the right answer, and if, after the verification of the department’s existence, which is the request allowing the system to react, a database request is built to know if Mr Smith belong to it, this action can not be justified but by the interpretation that this existing department is a possible answer.

Coming back to the dialogue study, with solely a 'yes' answer, the user has to take control of the dialogue, and to reformulate his problem, i.e. that he wants to speak to a Mr Smith of this department. Such a dialogue does not respect the co-operativeness principles.

However, when an interpretation is chosen, the system is able to change its mind according to the consequences of its inferences on the context. Such a behaviour can be illustrated with the above example when the utterance U2 is negatively evaluated by the database search module, with the diagnosis that the French department does not exist for instance. The
dialogue manager tries to reinterpret U2, and as it can be viewed as a yes-no question, it creates a sub-dialogue in order to give the user another possibility to answer. This decision also depends on the unfolding of the dialogue because we avoid to put the user in a deadlock due to a misreading of the answer. This part is managed by the interactional component.

The example can be completed in the following manner:

\textit{U1} - I want to speak to Mr Smith.
\textit{S1} - There are several Mr Smith in our staff
\textit{(S1a). Do you know his working unit? (S1b)}
\textit{U2} - Is there a French department?
\textit{S2} - No, there is not.
\textit{U3} - and an English department?
\textit{S3} - Yes, ...

\textit{Figure 4: The dialogue after reinterpretation of U2}

The meaning of an utterance can be viewed as the effects it produces in the context as in (Bunt 1994). We agree with his point of view where language is seen as a means to perform context-changing actions. We can distinguish actions to interpret the user's utterance from actions to generate the system's reaction. The interaction management is very important when deciding this latter kind of action. Its role is to detect problems, such as input failure and model failure, to allow the dialogue manager to maintain the communication and keep a coherent, friendly, and natural dialogue. However, it does not intervene in the first step of the analysis that determines the actions of the system to construct an interpretation, as we have seen in the above example. For that reason, it is less pertinent to our current purpose and we will not detail it. We will focus our attention on the processes that are useful for this preliminary step. Firstly, the thematic interpretation manages the topics present in the dialogue. Topics are the matters under discussion, and the informational coherence proceeds from the links between these topics. We distinguish main topics which are different subjects of conversation, not connected among themselves by structural links, from subtopics which are refinements, details ... of main ones and hierarchically organised. And secondly, the communicative intention analysis computes the conversational intentions of the user. These intentions result from the social behaviour of an agent participating to a conversation. They correspond to obligations an agent must respect to maintain the conversational coherence, for example the obvious obligation to answer a question.

As in (Airenti, et al. 1993, Traum and Allen 1994), we separate the recognition of these intentions from the usual intentions, because they correspond to different behaviours and rely on different pieces of knowledge. On the other hand, the distinction between information and intention level is necessary, as shown by (Asher and Lascarides 1994) and (Moore and Pollack 1992) when criticising the discourse relations defined in RST (Mann and Thompson 1988). The two kinds of corresponding relations must coexist between utterances because they entail different reasoning, and it is not possible to deduce the one from the other. As we said above, we have chosen to combine these approaches, and thus to examine the user utterances from three points of view. The advantage to separate the three approaches, in addition to the fact that they use different reasoning processes, is the opportunity given to the dialogue management to backtrack on one of these analyses, without modifying the others.

As we have already argued, the meaning of an utterance proceeds from the combination of the results obtained by the three analyses. In particular, the interpretation relies on the topic in progress and the communicative intention and we will show how they are computed and how they combine.

2). The different processes

2.1 Thematic interpretation.

The thematic interpretation (Grau 1984) works on the informational content of the utterances referring to concrete aspects of situations. It takes charge of coherence verification, inferences productions to integrate a new content in the previous discourse content and topic progress. It does not compute the truth value of an utterance. The questions which are addressed are: is the utterance coherent with the preceding topics, which are the necessary inferences maintaining this coherence and what kind of topic evolution do they suggest? In order to solve these problems, we use pragmatic knowledge which encodes the cultural aspects encountered in concrete situations and renders the different topics the system is able to recognise. They are represented by a graph of schemata. Close to Schank's MOP's (Schank 1982), the model allows different levels of description (from a more
general point of view to a more precise) and thus, avoids redundancy and gives a flexible way to encode the situations. These schemata are hierarchically related and are described by slots whose values are references to concepts of a semantic network or other schemata; these values can be constraint by means of facets allowing us to adapt the level of description to the situation itself. The slots structure the situation from its causal and/or chronological description.

A top-down and bottom-up reasoning process is used to build the link between the current sentence and the representation of the preceding discourse. The top-down aspect consists in taking into account the current context and making obvious the topics that are the most likely to be developed in it. We will precise later how this selection is made because it is a function of the whole representation of the dialogue. The bottom-up process computes what schemata are to be triggered from the sentence, and tries to find a path in the graph between the most relevant one and the schemata belonging to the context, by inferring intermediate schemata from general world knowledge.

According to the existence and the composition of the path, different types of coherence may be found. When the sentence gives the value of a slot, it is a topic confirmation. When this occurs, but with one lower or upper level, it is a precision or a generalization. When a longer path is found between two schemata, it is a topic deviation. Finally, an absence of path implies a topic shift. In this case, the discourse coherence relies on shared concepts between the current sentence and one of the previous topics. If no coherence is found, we are in a cock-and-bull discourse and the new sentence is considered to introduce a new subject.

A tree that represents the topic evolution as well as the current topic is updated. This tree represents the structure of the discourse from the topic point of view and indicates how the topics have been introduced and developed. We find here the informational relations as mentioned in (Moore and Pollack 1992) that are for us CONDITION, RESULT and DESCRIPTION coming from the structuring of the world knowledge. They convey information to the plan manager to establish the user's intentions. We also mark the second kind of relations we have defined between topics, namely CONFIRMATION, SPECIALISATION, GENERALISATION, DEVIATION and SHIFT. These one will be used by the dialogue manager to build the structure of the dialogue and decide the actions to undertake. After interpretation, the propositional contents of the utterances are integrated into the instantiated schemata. This model represents all that has been said about the different subject, without taking into account the order in which those different pieces of information appear.

2.2. Communicative intention analysis

The communicative intention analysis relies on the recognition of speech acts, as it is rather usual in most dialogue systems following (Allen and Perrault 1980), and on the attribution of illocutionary functions.

The application decides of the speech acts that may possibly be encountered during the kind of dialogues we are interested in. So they are priori determined, via the examination of corpora for instance, and compose a fixed set. What we try to do then is to detect which one is used in a given utterance. In our work, this recognition principally relies on the semantic interpretation of what has been uttered⁴ and thus the speech act is rather close from the semantic of the main verb. As long as the semantic analysis is not modified, the recognised speech act will not be changed in the following interpretation processes. At this level of interpretation, we are treating what Grice (Grice 1968a, Grice 1968b) calls the conventional implicatures, which are linked to the application.

To take into account the Gricean conversational implicatures, we identify an illocutionary function, which is intended to capture the communicative intention of the user. These functions are classified as initiative ones (which initiate a new sub-dialogue inside the dialogue), or as reactive ones (which react to a preceding initiative). An initiative illocutionary function is associated to a set of possible reactive illocutionary functions. For example, an ask for confirmation is linked to a confirmation or an invalidation. To have a complete corresponding table, see (Grau, et al. 1994a, Grau, et al. 1994b). The fact that an illocutionary function is initiative or reactive assigns this role to the corresponding utterance. This role is a way to indicate how an agent is obliged to do something by the other.

Thus, at the opposite of the speech act, the illocutionary function depends on parts of context, such as the dialogue structure, that we will develop later, and also from the user's goal.

Inside a given utterance, both the speech acts and the illocutionary functions are identified using a

⁴The pragmatic interpretation of the utterances comes after the analysis of the literal meaning which computes a semantic representation based on conceptual graphs (Sowa 1984).
set of rules, in which the most important premises concern the semantic analysis, the state of the dialogue obligations and the system expectations. The expectations concern what the user is likely to say (and, of course not what he is obliged to do), regarding to current goals and plans. These rules are organised, so as to privilege the ones giving priority to the conversational coherence.

In the following figure is an example of such rules, which is used to interpret U1.

Rule R1:
If propositional content = informative and positive (assertion or question)
dialogue obligation = none
system expectations = no current user’s goal, a new demand is awaited
then speech act = interrogate
  illocutionary function = ask for information.

Figure 5 : Recognition of the speech act and illocutionary function

In some cases, it may happen that more than one rule applies, concerning different parts of the utterance, which may be rather complex. Then if the speech acts and illocutionary functions are not the same, the utterance will be decomposed in the dialogue structure.

2.3. Collaboration of the interpretations

To be able to preserve how the different initiatives are related to the corresponding reactions, we build a structured representation of the dialogue. This structure is inspired by (Roulet, et al. 1985). Inside this framework, we define an exchange as being the smallest dialogue unit (at least two utterances from two different speakers). We also need to introduce the notion of intervention, which may be simple or complex. When it is a simple one, the intervention is uttered by a single speaker, but it has to be decomposed in several elementary interventions when more than one speech act and illocutionary function have been recognised. A complex intervention is composed of an exchange, (recursively) followed or preceded by an intervention, and thus has more than one speaker. To be more precise, an exchange is made of two interventions (an initiative one and a reactive one), potentially followed by a third one, which has the role of an evaluation. When an exchange follows an intervention, it is labelled as complementary, and when it precedes the intervention, it is labelled as preliminary, otherwise it is called a main exchange. The whole dialogue is then composed of a list of main exchanges. To have a more detailed presentation of this structure, the reader may consult (Grau, et al. 1994a, Grau, et al. 1994b).

To determine the exact meaning of the intervention and the consequent actions on the context so as to maintain the whole coherence, the dialogue manager takes into account the speech act and illocutionary function determined as we just explained before, and the kind of the link between the current topic, and the previous one determined by the informational module.

To build this structure, the dialogue manager has to decide what exchanges have to be closed, if a new exchange has to be created, how to label it, etc. A new main exchange corresponds to a topic shift, and a preliminary or complementary exchange to a topic deviation. In order to define the role of each intervention, the coherence of the illocutionary functions has to be tested.

Maintaining both the informational and conversational coherence does not required to give preference to one of the aspects. It is rather based on a general view of the current context dialogue by the preceding rules, and gives the system a dynamic behaviour.

This point is also exemplified when forming the list of the most pertinent topics useful to the topic manager. This local current context relies on the dialogue structure and is constituted by all the topics related to not yet closed exchanges, plus the last topic, even if apparently closed. It excludes topics in exchanges not formally closed but belonging to a higher, closed exchange. Such cases arise when subdialogues have been initiated and one of the partners has changed his mind. The end of the exchange has never been made explicit.

The local context represents the topics on which the conversation may resume without explicit clues. We do not forbid the others, but their reintroducing into the conversation needs to make an explicit reference to them (for instance, "Let us come back to ...").

As we have already said, it is obvious that the interpretation phase is still not complete: the recognition of the user’s intentions, and how they interfere in the current plans has to be done. What we argued is that the determination of the thematic and conversational context of the current utterance will serve to reduce the potential search space of these modules.
2.4. Application on the examples

Let us come back to the examples to detail the behaviour of the system when interpreting user's utterances.

First example:

U1 - I want to speak to Mr Smith.
S1 - There are several Mr Smith in our staff (S1a). Do you know his working unit? (S1b)
U2 - Is there a French department?
S2 - Yes, but Paul Smith from the French department (S2a) is not here (S2b).

U1 has led the system to settle the user's goal and to elaborate a plan that will allow to reach its main goal, i.e. to give the user an answer. As the database finds a lot of answers, the system5 conveys this piece of information to the user and finds a good criteria to choose among all the possibilities6.

When U2 arrives, the current context contains the user's goal, the topic, 'be member of a laboratory', the structure of the dialogue, the system's goal and the state of its plan where an obstacle has been detected. As we have already said, the diagnosis of the thematic analysis is a topic confirmation. An obligation to answer is settled, so the user's communicative intention leads the system to interpret U2 as an answer (rule R2, fig. 5). The dialogue structure is updated, with the prevision to give an evaluation in reply (application of the rule R3 fig. 9).

The database search is triggered with the query to find Mr Smith of the French department. The answer is positive then the dialogue manager sends the message to the planner which can follow its initial plan. The interpretation of U2 also results in obligations for the system to communicate an evaluation. It must explicit the inferences it has made (Mr Smith belongs to the French department), to respect the Gricean maxim of quantity (the complete identity of Mr Smith that was a problem to solve) and respect the conversational principles (formal answer to the syntactic question). Otherwise the only answer would have been Mr Smith is not here.

Second example:

U1 - I want to speak to Mr Smith.
S1 - There are several Mr Smith in our staff (S1a). Do you know his working unit? (S1b)
U2 - Is there a French department?
S2 - No, there is not
U3 - and an English department?
S3 - Yes, ...

The analysis progresses in the same way till U2 and more precisely till the database search when interpreting U2. But in this case, the database module diagnoses that the French department does not exist7. The dialogue manager decides to try to reinterpret the communicative intention of U2, before trying to elaborate a new strategy. Remember that the choice of the strategy relies on the dialogue ongoing evaluation. The dialogue manager gives a second chance to the user, with the agreement of the interaction manager to avoid to change its strategy. More generally the system keeps the same strategy as long as possible to improve the general dialogue coherence (topic changes in each turn-taking are sometimes difficult to follow even if they are coherent from the informational point of view).

The communicative intention analysis will find another interpretation of U2, as a yes-no question. This new interpretation is possible according to the syntactic form of the utterance : a rule, with lower priority due to the pragmatic prevalence, is triggered. The new illocutionary function is a request. The dialogue manager modifies the structure of the dialogue.

The reaction of the system (whose illocutionary function is negative answer) is integrated as the closure of this exchange (rule 2 fig. 8).

With the utterance U3, the context becomes similar to the one of the first case, and U3 is interpreted as an answer to S1b.

5It is not interesting to detail here the system working in term of the exact triggered processes, thus we only mention its global behavior.

6The best criterion is the kind of knowledge that mostly reduces the ambiguity. The formulation of the question about this criterion depends on the number of possible values. Here, there are too many working units to propose their list to the user (maxim of quantity), so the system chooses an open question (Vilnat and Sabah 1985).

7We can note that if the database module answer is that there is no Mr Smith in the French department, the dialogue manager also reinterprets U2, however, its answer will be more complete thanks to the first interpretation because it is now able to say that the department exist and that there is no Mr Smith in it.
Conclusion

We have shown that the utterance meaning in a dialogue depends on three interpretations, namely informational, communicative and intentional levels. Even if the informational and intentional interpretation are important, the communicative interpretation is central in a dialogue system. Each utterance must be interpreted in terms of the information and the intentions it conveys, but also in terms of its communicative role. We argued on the importance to differentiate each analysis, and then to mix the results to obtain the whole interpretation. We have especially underlined the topic and communicative intention analyses to explicit the first phase of interpretation which greatly influences the further actions of the system and reduces the work of the goal and plan modules, by handling some coherence problems.

References


Asher Nicholas and Alex Lascarides 1994, Intentions and information in discourse, Proceedings of 32th ACL, New Mexico State University, Las Cruces.


Grau Brigitte 1984, Stalking coherence in the 'topical' jungle, Proceedings of International conference on the fifth generation computer systems, Tokyo.


Grice Paul 1968a, Logic and conversation, Mimeo.


Grosz Barbara and Candace Sidner 1986, Attention, intentions and the structures of discourse, American journal of computational linguistics, 12, 3, pp. 175-204.


Mann William C. and Sandra A. Thompson 1988, Rhetorical structure theory : toward a functionnal theory of text organisation, Text, 8, 3.


Schank Roger 1982, Reminding and memory organisation : an introduction to MOPs, Strategies for natural language processing, Lawrence Erlbaum, Hillsdale, N.J.


Vilnat Anne and Gérard Sabah 1985, Be brief, be to the point, be seated, or Relevant responses in man-machine conversation, Proceedings of 9th IJCAI, Los Angeles.
TUTORING VERSUS TRAINING: A SPOKEN LANGUAGE DIALOGUE MANAGER FOR INSTRUCTIONAL SYSTEMS

Susann LuperFoy

The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102 USA
luperfoy@mitre.org

ABSTRACT
We propose an architecture for integrating discourse processing and speech recognition (SR) in spoken dialogue systems. It was first developed for computer-mediated bilingual dialogue in voice-to-voice machine translation applications and we apply it here to a distributed battlefield simulation system used for military training.

INTRODUCTION
Our work collects discourse-level processing into a centralized discourse capability as part of a modular user interface dialogue architecture. Its use in a spoken dialogue interface to a distributed battlefield simulation system used for military training is diagrammed in Figure 1.

The simulation system is called ModSAF or Modular Semi-Automated Forces. It runs on a network of Sun workstations viewed by instructors and students during mission rehearsal exercises. The user (a commander in training) issues spoken orders in a restricted subset of English called "battle command language." Commands are directed to platoons and companies represented in the simulated world.

The functional interface to ModSAF is through the "Command and Control Simulation Interface Language" or CCSIL (Salisbury, 1995) into which all spoken input utterances must be translated. DM converts the output of a speech recognizer into well-formed CCSIL commands, and presents backend responses to the user. It intercept errors that originate with user behavior or with any component of the input interpretation system. The Discourse Processor (DP) maintains a simple record of the developing linguistic context (Discourse State), and associates new utterances with it and with stored representations of context in the KB: (1) a User Model, (2) a static Domain Model containing rules for engaging the backend system, with a grammar for the language of well-formed, executable commands, and (3) a dynamic Backend Model (BEM) that maintains updated status for salient aspects of the non-linguistic context.

1.2 HC Dialogue and HCH Dialogue
Real-time spoken dialogue systems are of two types: human-computer dialogue systems or HC (for "Human-Computer") systems, and computer-mediated human-human dialogue systems or HCH (for Human-Computer-Human) systems. In HC systems, the system as a whole is engaged with the user in a dialogue, as in query processing for database retrieval or voice-activated command and control. In HCH systems, the computer only facilitates a
dialogue between two humans and is not a party in the dialogue. HCH systems include multi-user dungeons known as MUDs and voice-to-voice machine translation systems (Yamazaki et al., 1994), (Wahlster, 1993). (Waibel, 1995), (Montgomery, 1995).

HCH systems tend to require of the NL (Natural Language) interpreter shallower processing with broader coverage of the input language. In contrast, HC dialogue systems require deep interpretation of a narrower set of input commands. That is, if the user says, “Get one of them and take it to her” in an HC dialogue, the interpreter must fully resolve each reference, “one,” “them,” “it,” and “her” in order to locate the intended wrench and deliver it to the intended human being. The same utterance to an HCH system requires only sufficiently deep interpretation for transmission to the target language correspondents for “one,” “them,” “it,” and “her.” In HCH dialogue the system has no power to influence the breadth of language produced by the users, whereas HC dialogue can be designed as a fully system-initiated interaction in which the user responds only to options presented in a restricted vocabulary. In HC dialogue we can rely on human cooperation and adaptability to limit the input vocabulary especially when it corresponds to the limited functionality of the application task.

1.3 Tutoring Versus Training

Differing pedagogical objectives give rise to two modes in instructional HC systems. In “training” mode, the interface is transparent to the user who has the sensation of interacting directly with the backend application. The DM may simply record commands and responses in the training dialogue history, flagging trouble spots for retrospective analysis by an instructor. In contrast, the DM of a “tutoring” system, acts as a third-party mediator whose job includes intercepting user mistakes for correction before passing them to the backend simulation system.

Consider a student’s mistaken command to the first platoon to increase its speed by 30 kilometers per hour which would result in an illegal (impossible) velocity. A tutoring system could interrupt the primary command-and-control dialogue between the student and the simulation system to say: “The maximum velocity for a platoon is 60 kph. First platoon is already traveling at 40 kph.” In contrast, a training system would send the illegal command to the backend system without modification and leave it to the user to recognize that the wrong platoon was invoked or a problematic velocity value assigned. Our goal is a DM that can be set to operate in either training or tutoring mode by simply adjusting the parameters that control its reaction to student mistakes.

2. DIALOGUE MANAGEMENT

In a deep+narrow HC dialogue, errors can be prevented by designing the NL interface to interpret only those utterances that have meaning to the system and constraining the user’s input language to match that narrow sublanguage. It is important to present to the user reliable indicators of the system’s restricted language processing ability and avoid an “Eliza-styled” interface that appears superficially to have much greater understanding than it actually does, leading users to say things that cannot be understood in terms of the backend command language. When prevention fails we rely on the recovery mechanism.

The architecture for our work on spoken dialogue interfaces to distributed military simulation systems (Figure 1) was first developed for computer-mediated (HCH) bilingual dialogue in voice-to-voice machine translation applications. The DM is a third-party mediator translating between user and backend languages. Before “translation” from English into the language of the backend system, the DM invokes its recovery algorithm to evaluate the input utterance interpretation against the non-linguistic context representations to do preemptive troubleshooting.

2.2 Dialogue Trouble

Temporary miscommunication in computer-human dialogue (Terveen, 1991) is not an artifact of today’s incomplete technology; but an inevitable feature of dialogue interaction. We use Clark and Schaefer’s (1987) analysis of human human dialogue as a starting point for the development of a four-step process for dealing with disfluency in HC dialogue. The four steps are Detection, Diagnosis, Repair Plan Selection, and Collaborative Plan Execution. (Prevention is a fifth category of effort that we treat as external to but supportive of the recovery process proper.)

In their study of human-human dialogue, Clark and Schaefer observe a process of grounding by which participants work jointly to ensure listener
“acceptance” of each new speaker “presentation” of material for incorporation into the common ground (i.e., the shared understanding) of the discourse. Brennan et al. (1993) and Perez-Quiones (1996) formalize this process for typed database retrieval dialogues so that the human user receives an indication from the system of the internal status of the query interpretation process. Traum and Allen (1992) apply a similar grounding model to the analysis of a corpus of collaborative problem-solving dialogues between humans. In our system, the grounding or acceptance phase serves as the point of attachment for repair dialogues. We expand the acceptance phase into a 4-step repair-management algorithm invoked for each input utterance.

2.3 Levels of Interpretation Failure

Approximating Clark and Schaefer’s eight levels of understanding in human-human dialogue we define eight categories of communication failure in our architecture.

Level 0: The dialogue manager (DM) notices that no speech was received following a prompt to the user.

Level 1: Speech recognition component (SR) reports below-threshold confidence or cannot return any hypothesis at all for the current acoustic input.

Level 2: Sentence interpreter cannot parse the current utterance.

Level 3: Sentence interpreter cannot assign a semantic value to the current utterance in order to translate it to a well-formed CCSIL command. For example, “FIRST PLATOON INCREASE SPEED” is missing an argument for the velocity feature value and in “FIRST PLATOON CHANGE FORMATION TO THREE FIVE KILOS.” there is a semantic type conflict.

Level 4: The utterance translates to a well-formed command in the backend language but execution of that command would result in the violation of a constraint in the Domain Model. For example, “FIRST PLATOON INCREASE SPEED TO 400 MILES PER HOUR” is well-formed but it violates the maximum velocity constraint of 60 kph.

Level 5: The command is well-formed and legal but infelicitous in the current discourse context due to a failed precondition in the Backend Model (BEM). For example, “FIRST PLATOON DECREASE SPEED BY TWO ZERO (20) MILES PER HOUR” results in a Level 5 failure if the current velocity of that platoon is only five miles per hour since execution of that command would result in the violation of a range constraint.

Level 6: The backend system rejects the input command for reasons that are beyond the DM’s understanding of the backend system as represented in the Domain Model. In other words, our check against the Domain Model and BEM was insufficient to detect a problem.

Level 7: User notices that the backend application’s response fails to satisfy the user’s objective. Consider the case where the user says “FIRST PLATOON HALT” but the SR component returns “THIRD PLATOON HALT” albeit with a high confidence score. That error may go undetected through Level 6 if first platoon is in motion so that the BEM meets preconditions for halting. In this case, the system issues a command to the simulation to halt the wrong platoon.

2.4 Steps in the Recovery Algorithm

The generic four-step error recovery algorithm we posit for any spoken dialogue system begins when either of the dialogue participants detects an error condition.

1. Detection: In spontaneous dialogue, miscommunications can go undetected by both participants indefinitely. may discover years after the conclusion of a dialogue that I had misunderstood something said to me, or I may never detect the miscommunication. In automated dialogue systems, the SR, parser, or DP can fail to detect its own misinterpretation. If the problem goes undetected by subsequent interpretation processes, the backend application program fails to respond as the user intended and it is the user who must detect the error condition.

2. Diagnosis: Once an error has been detected it is classified into Level 0...7. Detection with no ability to diagnose can an interface that simply requests, “PLEASE REPEAT,” regardless of error type, which is neither efficient nor pleasant as an interaction partner. Level 4 errors are detected by the Dialogue Manager with the help of a stored Domain Model which holds the knowledge base representation of constraints that backend system designers wish to have enforced by the user interface. Level 5 errors are detected by the dialogue manager after consulting the Domain Model and BEM. For a detected Domain Model violation, the actual cause could be the user misconception or slip of tongue, SR error, or interpretation failure.

3. Repair Plan Selection: Given a diagnosis, the system has flexibility in selecting a repair plan. In
tutoring mode, the dialogue system usually hopes to catch any student misconception, and engage the student in a dialogue to correct the flawed beliefs or reasoning. But depending on the objectives for the immediate tutoring session, it may instead make sense to simply fix this error without notification, to avoid distracting attention away from the primary objective. Given the identical diagnosis, a training system might select a different repair plan to let the student experience the (simulated) consequences of certain errors. A well-formed but disastrous command can be admitted so that the student commander inadvertently destroys a friendly platoon. This experience of negative consequences is a key motivation for the use of simulated training environments.

Finally, when an interpretation module detects an error the design decisions may allow the DM to repair the miscommunication unilaterally, in the process of converting the utterance into a command, without bringing it to the user's attention. A command to the simulation to reduce by 20 mph the velocity of the a platoon traveling 5 mph, causes DM to halt the platoon and inform the user.

4. Execute the plan interactively: The human participant introduces unpredictability during the repair dialogue just as in the main dialogue. This means that execution of the plan constructed in Step 3, must be flexible in the face of a user's failure to conform to the selected plan. This step requires, in the limit, a full-blown reactive planning program. As an approximation we can implement a system that allows for clarification questions where the plan structure predicts a yes/no response to a repair question. We have thus far implemented no plans that grant flexibility to the user so we have little to say about this step.

3. CONCLUDING REMARKS
This paper reports on an in-progress implementation project in which our dialogue manager (DM) supports a spoken dialogue interface to a distributed battlefield simulation system. The DM acts as the mediator in a mixed initiative dialogue, interpreting between one user and the simulation application. For the purpose of this discussion, we have focused on the DM dialogue repair procedures, ignoring (a) the anaphora resolution mechanism, (b) updating routines that incorporate a new input or output utterance and its interpretation into the Discourse State, (c) the plan recognition component, and (d) processing of output utterances from the simulation system to the user. We have yet to address several difficult problems engendered by the multi-user distributed nature of the training system and treat only the dialogue management problem for a single-user interface to the distributed simulation in which other processes (human or machine) may affect the state of the represented world.

It is important to acknowledge that all four steps to our recovery algorithm involve uncertainty, and therefore risk. It may be impossible to determine whether a given unexpected utterance received by the DM is the result of a user misconception, a speech recognition error, or a weakness on the part of the DM itself; i.e., the Domain Model could be missing crucial knowledge about the backend system, DP may have failed during anaphora resolution (interpretation of context-dependent noun phrases), DM may have failed during error detection or diagnosis at some point in the dialogue history leading to a corrupted discourse context representation, etc. Therefore, what is most interesting and potentially fruitful in this enterprise is the discovery or development of reliable heuristics for carrying out the four steps. What we have provided here is an architecture into which those heuristics and rules can be inserted, and coordinated with other centralized discourse functionality.

For further work on the integration of discourse processing and speech recognition, we are looking at uses of Discourse State information to select among multiple alternatives (n-best) hypotheses produced by SR. Secondly, we want to allow the DM to alert SR in advance with a prediction for which speech act is expected next and therefore, which of a set of pre-trained language models to select as the language model for the next input utterance or which certainty threshold to adopt. Finally, we are negotiating a mechanism for SR to preserve and transmit to the discourse processor, prosodic data of value to discourse processing but not currently used by SR or NL sentence analyzer.

4. REFERENCES


BUILDING ON EXPERIENCE:
MANAGING SPOKEN INTERACTION
THROUGH LIBRARY SUBDIALOGUES

David G. Novick & Stephen Sutton
Center for Spoken Language Understanding
Oregon Graduate Institute
P.O. Box 91000, Portland, OR 97291-1000 USA
{novick, sutton}@cse.ogi.edu

ABSTRACT

Management of spoken dialogue systems is a difficult process to abstract because it closely reflects the domain task, relies on specific task contexts to constrain user actions, may depend on poorly understood dialogue control acts, and often requires that system prompts be hand-tuned. Construction of dialogue managers is also difficult, because interface-design tools do not embody useful knowledge of experienced developers. These problems can be largely addressed through the development and use of libraries of subdialogues that represent the familiar subtasks of everyday conversation. We present such a library of subdialogues and describe two prototype spoken dialogue systems constructed from subdialogues in the library. The library enables developers to deploy tested elements of a repertoire of human-machine interaction that build on experience.

1. INTRODUCTION

While tools and environments for building spoken dialogue systems (SDSs) demonstrably speed up the development process, authors using these tools still produce applications on a custom or “one-off” basis. With each system they generally start from scratch, encountering problems in dialogue management that others may already have solved through domain expertise and hand-tuning.

Tools for building SDSs should incorporate this experience into the resources offered to the tool’s authors. One way of doing this is to provide libraries of subdialogues that incorporate useful, tested means of spoken interaction. From these subdialogues, authors should easily be able to construct spoken dialogue systems for more complex, domain-specific tasks. This approach is particularly useful because the nature of dialogue management often depends on the particular domain or task addressed by the interaction.

In this paper, we suggest that dialogue management should be considered important at the subtask level. We review reasons why dialogue management has been a difficult problem, and why authoring tools for SDSs typically produce unusable systems, particularly when the author’s expertise lies in a domain other than spoken-language understanding systems. We propose, then, that these problems be addressed through development and use of libraries of spoken-language subdialogues that represent the familiar subtasks of everyday conversation. From subdialogues in these repertoires whole applications can be built.

To demonstrate the usefulness of subdialogue libraries, we developed and deployed a collection of initial subdialogues in a rapid-prototyping tool for SDSs. Section 4 describes the contents of the library. From elements of this library, we built prototype SDSs, with some subdialogues in common. Section 5 describes two such systems. We also discuss problems that arose in the development of the libraries and prototypes and suggest avenues for future work.

2. PROBLEMS OF SPOKEN DIALOGUE SYSTEMS

Dialogue management for a task-based spoken-dialogue system enables the user and system to meet the user's goal by accomplishing a task and provides useful constraints on dialogue context so that both user and system have better expectations about their respective actions. Dialogue management functions include interpreting utterances in context, deciding on the next action to take, constraining responses, and repairing conversational breakdowns. To do this, a dialogue system maintains context, tracks goal structures and determines levels of acceptance, supports
coherence across turns, and possibly manages initiative.

How to provide these functions effectively is an open question. Many dialogue management systems have been proposed for spoken interaction. The evidence suggests that the most effective systems are those based on experience in understanding the particular contexts of interaction that occur in the dialogue. This relative failure of abstraction is the result of task-related difficulties in dialogue management. Moreover, the nature of these difficulties makes developing SDSs correspondingly difficult, particularly for non-expert developers.

2.1. Difficulties with Managing Dialogue

For spoken dialogue systems, the challenge of dialogue management is to provide convenient access to relatively complex information using relatively simple interfaces devices such as the telephone (Sparks, Mclntosh & Bruner, 1994). This challenge is symmetric, in that SDSs face like difficulties in obtaining information from users, without which the SDS cannot take meaningful action on the user's behalf. In particular, we can identify four sources of difficulty in managing spoken dialogues: (1) the distinction between managing the domain task and managing the dialogue is not clear, especially because task-oriented dialogue has been shown to reflect the structure of the domain task; (2) creating contexts that clearly communicate expectations is hard; (3) processes like initiative and acceptance are poorly understood; and (4) system prompts often need to be hand-crafted.

The connection between the structures of dialogues and the tasks they embody was described by Grosz and Sidner (1986). Consequently, the control structures of SDSs typically reflect the domain action structures of the tasks they implement. For example, the main structures of the voice navigation system developed by Sparks, Mclntosh and Bruner (1994) closely reflected domain task elements at different levels of abstraction. Similarly, the design of an intelligent answering machine system (Ludovik & Sibirtsev, 1995) not surprising tracks the domain functions of a secretary. The design of Larsen's (1995) automated book club service closely tracks five key tasks. In all of these examples, the system's dialogue manager has the responsibility for carrying out the domain task itself, such as guiding the user to his or her intended destination. This suggests that dialogue management, in a general manner abstracted from particulars of the domain, is either infeasible or intractable.

In managing the domain tasks, the second source of difficulty for SDSs is that they need to shape the users' utterances in such a way that these utterances are most easily interpretable. Knowing how to do this in a way that is not discouragingly heavy handed is an open problem (see Zoltn-Ford, 1991; Patrick, Jacques-Lomelis & Whalen, 1993). The issue is how to create contexts—of domain actions, control actions and language—that situate the user in such a way that he or she produces utterances that have characteristics of action and language about which the system can form reliable expectations. Reports of developing SDSs, in examples such as (Zue et al., 1993; Cole, 1994), suggest that building such contexts is more a matter of art and experience than an exact science. Zeigler and Mazor (1995) found that an engineered system for telephone disconnect orders required a recognition vocabulary an order of magnitude smaller than that predicted by estimation of parameters from human-human conversations in the same domain.

Third, just as the structure of the interaction relates to the structure of the task, so too do control characteristics of interaction depend on domain elements. Reports from builders of prototype SDSs indicate that felicitous conversational initiative is often a function of special knowledge about the domain. (van der Hoeven et al., 1994; Fanty et al., 1995). This suggests that abstraction of mixed initiative for SDSs is likely to be impractical for the foreseeable future. Rather, changes in initiative will have to depend on hand-crafted engineering of the dialogue manager with respect to the particulars of the domain. Likewise, the issue of how best to perform functions of conversational acceptance (Clark & Schaefer, 1989) also appears to require hand-coding in order to achieve reasonable effectiveness of interaction. While we have argued (Novick, 1988; see also Traum & Hinkelman, 1992) that spoken interaction can be characterized by speech-act-like constructs at many simultaneous "levels" of conversation acts that are fundamentally domain-independent, this line of research has yet to produce a set of operators for acceptance that are robust enough to be deployed in SDSs in real domains.

Fourth and finally, creating the linguistic contexts required for effective spoken dialogue (see Brownsey, Zajicek & Hewitt, 1994; Cole et al., in press), despite progress in characterizing the space of possible prompts, remains a matter of heuristics for formulating specifically appropriate prompts within the design space (Hansen & Novick, 1996). Consequently, the normal practice of development is to design several versions of the system, each benefiting from empirical evaluation of the preceding versions (Fraser, 1995). Thus development of practical SDSs typically requires either tuning of the dialogue model via multiple iterations of a design-and-test cycle.
involving interactions with users (see, e.g., Baekgaard et al., 1995; Cole et al., in press), or painstaking analysis on a case-by-case basis of the context created by the system's prompts (cf., Takano et al., 1993), or both.

These four difficulties in managing dialogues have a common thread: the need to customize the dialogue for the particulars of the task and domain. The most effective SDSs appear to result from the application of experience rather than abstraction.

2.2. Difficulties in Constructing SDSs

A separate thread involves the methods through which spoken dialogue systems are constructed. Schematic or direct-manipulation systems for building graphical user interface (GUI) systems, especially for non-expert users (e.g., Douglas, Doerry & Novick, 1992; Pausch, 1991), are being mirrored by a generation of interface construction tools for SDSs (e.g., Larson & Baekgaard, 1994; Fraser & Thornton, 1995; Sutton et al., in press). These hold the promise of being able to develop SDSs much more rapidly.

However, even using systems for prototyping SDSs, we can expect that developers—especially non-experts—will have difficulty in creating effective interfaces. Many researchers in human-computer interaction believe that rapid prototyping systems for interfaces primarily enable people to develop terrible interfaces faster. A survey of software developers indicated that, for many of the respondents, the most difficult problems arose of the design of the user interface rather than its implementation (Myers & Rosson, 1992). This is likely to be equally true for spoken-language systems as it is for GUIs.

Novice (and even expert) developers particularly stumble in building SDSs due to a mismatch between (a) our procedurally oriented approach to thinking about computer systems and (b) the multiple layers of complex procedures required for effective spoken interaction. Conversation is comprised of a large number of subtle processes, many of which pass unnoticed in normal dialogue; it is unreasonable to ask novice authors to account for these. Alternative approaches using declarative specification of outcomes, such as goals composed of beliefs, fare no better, as they seem too technical, unwieldy, and weak, especially for use by non-specialists.

As a result, SDS builders that simply present the user with a palette of recognition and output functions are likely to strand authors in a confusing marsh of systems that underperform. It would be much better somehow to include in the SDS builder as much expertise as possible about the engineering of spoken interaction. This expertise should encompass the experience that others have gained in building systems with similar functions.

3. REPERTOIRES OF INTERACTION

Significant parts of the problems of managing dialogues and building spoken dialogue systems can be addressed through the use of libraries of subdialogues that represent common conversational tasks and subtasks. The subdialogues provided by the SDS builder should encapsulate experience from the standpoint of both (a) the results of experience in engineering spoken interaction and (b) the underlying experience of known conversational situations. Indeed, our view is that subdialogues represent pieces of interaction that people find familiar from reuse in everyday life. That is, much conversation can be constructed from parts of repertoires of interactions that people carry with them; speakers of a language know how a dialogue about, say, asking for the time of day should proceed. Following this intuition, much of a typical SDS can implemented by composition of subdialogues, providing already well-engineered interaction to the extent possible.

Through the use of library subdialogues, dialogue management can be constructed in a way that (1) naturally reflects the structure of the domain task because it is created through composition of domain subtasks, (2) provides contexts that are known to communicate expectations clearly to users, (3) incorporates domain-specific implementations of dialogue control processes such as initiative and acceptance, and (4) provides pre-tuned system prompts.

As a consequence of this approach, subdialogue libraries should improve the efficiency of building good SDSs by taking advantage of reuse. In most current development of SDSs reported in the literature, each new system is a "one-off" effort. With reuse of library subdialogues, authors can avoid starting from scratch, take advantage of well-constructed interaction fragments, and develop by assembly rather than by detailed programming.

Library subdialogues also spare authors from dealing with low-level technical details and provide guidance through examples. This should be particularly valuable where subdialogues are defined using a visual programming language in which the flow of control is simple to understand.

An important aspect of building SDSs from subdialogues in a widely available platform-independent environment is that developers will be able to share new subdialogues and libraries. To the extent that par-
ticular subdialogues achieve ubiquity, reuse of subdialogues can improve effectiveness of end-user interaction by establishing canonical interaction patterns as a community expectation. That is, the common use of an SDS subdialogue can help to add new elements to users’ repertoires of interactions.

4. A PROTOTYPE SUBDIALOGUE LIBRARY

To explore the usefulness, practicality and power of library subdialogues in addressing the problems of managing dialogues and building SDSs, we have developed an initial library of canonical subdialogues from which applications can be assembled. The subdialogue and the applications built from its subdialogues were all developed with the CSLU Rapid Prototyper (CSLUrp, pronounced “slurp”) (Sutton et al., in press). The subdialogue is one of the primitives in CSLUrp’s development environment. Consequently, CSLUrp enables authors to create new subdialogues that can be added to the library, thus increasing its coverage.

In this section, we briefly introduce CSLUrp, review the ways in which CSLUrp supports subdialogues, and present the libraries of subdialogues and related functions that we have developed in, and as a part of, CSLUrp.

4.1. CSLUrp

CSLUrp is a graphically-based authoring environment that incorporates all the steps necessary for building and executing simple spoken-dialogue systems. The main strengths of CSLUrp include: (a) the speed with which application prototypes can be created; (b) an easy-to-use interface; (c) strong support for authors who lack specialized technical expertise in speech recognition; (d) the ability to create a wide range of real-world applications; and (e) suitability for a broad community of users.

CSLUrp includes a graphical palette of dialogue objects and a simple drag-and-drop interface. The dialogue objects serve as visual-programming building blocks. During the design phase, the author selects and arranges appropriate objects, linking them together to create a finite-state dialogue model. Then, during the run phase, CSLUrp provides a real-time animated view of the dialogue. The author can alternate between the design and run phases, enabling the incremental development and iterative refinement of spoken language systems.

The set of objects in the palette covers a range of fundamental spoken language system functions including answering the telephone, speaking a prompt, recording speech input, recognizing speech input, and identifying DTMF tones, and invoking a subdialogue.

The interface is designed to require minimal technical expertise on the author’s part and to simplify the design and specification process. For example, specifying a speech recognizer is largely automated; all that is required of the author is to enter the recognition vocabulary by typing or saying—for speaker-dependent recognition—the target words. The text of prompts can be generated automatically according to a set of pre-specified styles (Hansen & Novick, 1996).

CSLUrp is implemented on top of CSLUsh (pronounced “slush”) (Schalkwyk, Colton & Fanty, 1996), a programming shell made up from a collection of core libraries, written mostly in C. The libraries can also be used without the shell for developing stand-alone C applications and third-party applications. Basic functions include manipulating wave files, performing signal analysis (e.g. FFT, mel cepstrum), extracting features, training and utilizing artificial neural networks, and doing speech recognition for isolated words and continuous speech with finite-state grammars. It includes a general-purpose (vocabulary-independent) recognizer and a number of special-purpose recognizers for common vocabularies such as digits and alphabets.

4.2. Using subdialogues in CSLUrp

Dialogue libraries in CSLUrp are implemented in terms of the subdialogue object, represented as a distinctive icon in the palette and in the system being built. Figure 1 shows the subdialogue icon in both the palette and in the system being constructed. Libraries are selected and loaded in the same way as regular files. Once loaded, the library appears as a single subdialogue object. This can then be expanded to display its contents as a separate window. Because dialogue management functions are built into each subdialogue, authors are largely relieved of the burden of specifying the dialogue-management component of their application; these functions can be customized by the creators of subdialogues to reflect the particular requirements of the domain subtask, such as formulating options or handling mixed initiative.

There is no restriction on the structure or content of library subdialogues. They can contain anything from a single state to a nested collection of subdialogues. In fact, a subdialogue can even be defined in terms of other subdialogues. States inside a library subdialogue can perform a wide range of operations such as
speech recognition (general-purpose or task-specific) and can execute Tcl or CSLUsH actions.

Once loaded, library subdialogues are much the same as a regular (i.e., developed by the author) CSLUr p subdialogue. This means that the author is free to modify them as required. Library subdialogues can require minor alterations to the system prompt. For instance, the library prompt "What city?" might need modifying as "What departure city?" This, however, is extremely simple because CSLUr p supports text-to-speech synthesis.

Subdialogues support the passing of parameters when called. This is a convenient feature for libraries in that it can be used to set up task-specific details. For instance, the password library object might include as arguments a list of users and their respective passwords. CSLUr p can also save individual subdialogues and their contents. This enables authors to extend CSLUr p and to develop their own subdialogue libraries.

4.3. CSLUr p Library

CSLUr p currently contains a selection of library subdialogues but these represent only a fraction of the possible set of useful interactions. So far, our efforts have involved creating new library subdialogues as the need arises; the number of library subdialogues can be expected to grow as we continue to develop additional prototypes with novel requirements. Some of the subdialogues currently available in CSLUr p’s library include:

- **Time-of-day.** This subdialogue queries the user for a time of day. The recognition vocabulary includes phrases such as, “ten thirty,” “ten fifteen p.m.,” “noon,” “midnight.” It could easily be extended to include other non-specific and relative temporal expressions such as “early,” “mid,” “late,” “morning,” “afternoon.”

- **Day-of-week.** This subdialogue queries the user for a day of the week. Recognition vocabulary includes “Monday” through “Sunday,” as well as common relative phrases such as “yesterday,” “today,” “tomorrow.”

- **Day-and-time.** This subdialogue queries the user for day and time information. It is defined in terms of the time-of-day and day-of-week library subdialogues. Figure 1 shows the content of this subdialogue and how it is composed of other, nested subdialogue elements.

- **Day-and-time-period.** This subdialogue queries the user for day and time period, made up from a start day/time and an end/day time. This is defined in terms of the day-and-time library subdialogue.

- **Password.** This subdialogue queries the user for their password. It will allow several attempts before failing. A list of users and their respective passwords must be supplied as an argument. Returns a pass or fail flag depending on whether the password was recognized correctly.

Other library subdialogues that we have implemented in CSLUr p include:

- **Date,**
- **Phone-number,**
- **City,**
- **Social-security-number,**
- **Zip-code,**
- **Spoken-name,** and
- **Spelled-name.**
The library mechanism can also be used to implement commonly performed tasks other than interacting with the user. This feature is extremely useful for simplifying the design procedure by providing high-level functions that perform complex tasks. The CSLUrp library currently contains these functions, available as subdialogues:

- Current-clock. This function returns current date/time information including year, month, day, hour, minute, second, day of week and total elapsed seconds. The latter is a convenient format when comparing dates.
- Get-URL. This function retrieves a document from the World-Wide Web. Requires a URL as an argument.
- Record-wave. This function records audio input into a wave object using either microphone or telephone device. Optional arguments include maximum record time and utterance detection parameters. Returns wave object handle.
- Play-wave. This function plays a wave object using either telephone or speaker audio device. Requires wave object handle as an argument.
- Words2numbers. This function converts between a digit string (e.g., “1234”) and a written number (e.g., “one two three four”).

As a logical extension of this hierarchical approach, the library supports objects of larger and larger granularity. For instance, building on the components we have already described, it is easy to imagine an appointment-scheduling library subdialogue that takes as an argument the current calendar of events and interacts with the user to schedule an appointment in an available time slot. In general, as the granularity increases so does the specificity of the task and domain. Thus, in time we expect to be able to provide a collection of domain libraries whose entries reflect the kinds of interactions and operations common in that particular domain.

5. BUILDING SYSTEMS FROM SUBDIALOGUE LIBRARIES

We have built a number of prototype spoken dialogue systems that use a selection of library subdialogues in order to demonstrate feasibility. The following examples—a call blocking service for a telephone line and an information service for railway schedules—illustrate both the functionality of the subdialogue components and the reuse of these components in different applications.

5.1. SDS for call blocking

This system allows the user to schedule blocking of incoming telephone calls by voice. In practice, the call-blocking mechanism would either redirect all incoming calls to the user’s voice-mail system, notify the caller that the user is unable to receive calls, or return a busy signal, at the user’s discretion. The implementation in CSLUrpf, using subdialogues, is depicted in Figure 2.

Our implementation operates in three basic call-blocking states: “unscheduled”, “scheduled” or “active”. A schedule becomes active only when the current day/time falls within the scheduled period. Scheduling call blocking involves specifying the day and time for both the beginning and end of the call-blocking period. Once scheduled, the system allows the user to reschedule or cancel.

Here is a sample dialogue that would be typical of the interaction between the call-blocking system (S) and a user (U). The parts of the interaction derived from library subdialogues are presented in roman font; non-library interaction is presented in italics.

S: Welcome to the CSLU call-blocking service. Please say your password.
U: Secret.
S: Call blocking is currently turned off. Do you want to schedule call blocking?
U: Yes.
S: What start day?
U: Today.
S: What start time?
U: Three thirty.
S: What end day?
U: Wednesday.
S: What end time?
U: Nine a.m.
S: Okay, your calls will be blocked from today at three thirty p.m. until Wednesday at nine a.m.

Good-bye.

Our implementation involved the use of several library subdialogues. In particular, it relied heavily on the day-and-time-period and day-and-time subdialogues for specifying and modifying schedules. It also included the password subdialogue for preventing unauthorized access, and it made use of the current-clock library function for determining whether or not the schedule was active.

Although the system is simple, it displays the basic functionality required for an end-application and could be easily extended. Getting this far required very little task-specific engineering of the library subdialogues used. Moreover, additional dialogue management functions were mostly of the nature of
connecting states together and expressing dialogue flow at decision points in terms of simple Tcl expressions. The most technical parts of the development process involved converting day/time information to and from a total-elapsed-seconds format and performing file handling of schedules, which are both relatively easy operations in CSLUs.

5.2. SDS for railway schedule information

Our next demonstration system is a simple information service providing railway schedules over the telephone. This task shares common ground with the call-blocking example in its need for day and time specifications. Its structured dialogue means that we were able to use many of the same library subdialogues, including day-of-week and time-of-day. The implementation in CSLUs, using subdialogues, is depicted in Figure 3.

Here is a sample dialogue that would be typical of the interaction between the railway schedule information service (S) and a user (U). As in the first example, the parts of the interaction derived from library subdialogues are presented in roman font; non-library interaction is presented in italics.

**S:** Welcome to the CSLU railway schedule service. Would you like information on departures or arrivals?

**U:** Departures.

**S:** From which station?

**U:** Schiphol.

**S:** To which station?

**U:** Amersfoort.

**S:** What day?

**U:** Wednesday.

**S:** There are 27 trains from Schiphol to Amersfoort on Wednesday. Approximately what time?

**U:** Around six-thirty in the morning.

**S:** Departure times include 5:57 a.m., 6:27 a.m., 6:36 a.m., 6:57 a.m. and 7:06 a.m. Do you need more details?

**U:** No.

**S:** Would you like information on another railroad service?
U: No.
S: Good-bye.

The system supports queries about the arrival and departure of trains. The user gives some approximate time information and the system presents the nearest train schedules in blocks of five. The user can then request details on any specific train such as its arrival time. Queries about specific trains also provide status information when available, such as whether a train is on time or delayed 20 minutes.

The system uses library subdialogues for from-city, to-city, day-of-week and time-of-day. Minor alterations were made to the system prompts in some of these subdialogues.

Dialogue management is relatively straightforward due to its structured design. Some basic inference capabilities are needed to interpret temporal information, such as "No later than seven."

6. DISCUSSION

While useful and usable SDSs can be developed with this approach, several significant problems of dialogue management remain to be addressed. These open issues include maintaining coherence of reference among subdialogues and ensuring that the top level of the application is well-constructed.

In constructing the initial set of subdialogues in the library, we used Tcl variables as the value-passing mechanism. In our first implementation, these variables are visible to all dialogues and subdialogues in the system being built, so there is a risk of collision among variable names, as well as a temptation to develop unstructured code. As more complex systems are constructed with CSLUrп, a safer implementation becomes increasingly important.

A systematic approach to inter-module coherence of reference can be developed from use of Grosz and
Sidner’s (1986) notion of the focus stack. This mechanism for keeping track of the point of the interaction does not correspond to a particular subdialogue; rather, a simple version can easily be implemented in Tcl that is consistent—and helps to assure consistency of reference—across subdialogues.

One way of ensuring that the top level of an application is well-constructed would be to develop libraries of entire applications along principles similar to those employed for the subdialogues. This approach begs the question, however, of providing guidance for authors of new applications. The fact that subdialogues are coherent is no guarantee that the overall application is likewise coherent.

This research also suggests a number of avenues for future work. The first and most pressing involves continuing to add to subdialogue library; as more researchers and developers use CSLUrp, the natural mechanisms of sharing and transfer should accelerate this process.

CSLUrp’s core functions will continue to grow, and the power of the library approach can be correspondingly extended. For example, CSLUrp should soon have an integrated semantic chart parser, so we will be able to extend the library to encompass semantic grammars, typically corresponding to linguistic requirements of the dialogue elements. We also plan to provide built-in support for use of a focus stack.

As is evident from the two prototypes presented in this paper, we have been working so far mostly with structured dialogues. Our research plans call for using CSLUrp and its subdialogue library to implement more flexible systems for tasks such as appointment scheduling (cf., Fanty et al., 1995). In this process, we expect to encounter a new set of problems, such as state-space explosion.

Finally, we anticipate being able to provide better support for repair. Currently the library provides a default back-off repair strategy when an individual subdialogue does not have a custom strategy. Repair will be addressed in two ways. First, different repair strategies can be included as meta-dialogue elements of the subdialogue library. Second, as we gain experience with use of the subdialogues in the library, the custom repair strategies can be tuned to reflect what we learn.

7. CONCLUSION

In this paper, we showed that it was possible to address problems of dialogue management and construction through libraries of subdialogues that incorporate specialized knowledge of subtasks. We described an initial library of subdialogues developed in the CSLUrp rapid-prototyping environment, and presented two prototype SDs that share common subdialogues. While the initial library is limited in extent, widespread use of CSLUrp holds the promise of building on developers’ experience through sharing of new subdialogues. As a method of building SDs, the subdialogue-library approach enables developers to deploy tested elements of a repertoire of human-machine interaction that build on experience.

8. ACKNOWLEDGMENTS

This research was supported by funding from U S WEST, the Office of Naval Research, the National Science Foundation, the Defense Advanced Research Projects Agency, and the members of the CSLU. CSLU staff who designed and implemented CSLUsh and CSLUrp include Ronald Cole, Mark Fanty, Pieter Vermeulen, Johan Schalkwyk, Jacques de Villiers and Azdine Tadrist. We thank Karen Ward for her efforts toward making this paper comprehensible.

9. REFERENCES


COMMUNICATIONAL DEVIATION IN FINALIZED INFORMATIVE DIALOGUE MANAGEMENT

Latifa TALEB
INRIA (Institut Nationale de Recherches en Informatique et Automatique)
Domaine de Voluceau, Rocquencourt
B.P.105, F 78153 Le Chesnay Cedex
E-mail: Latifa.Taleb@inria.fr

ABSTRACT

The aim of this paper is to propose a linguistic study of human "communicational deviation" which could be automatically processed so that it may be taken into account by a Human-Computer Dialogue system.

Our study is based on a corpus analysis. The corpus is composed of human-human task-oriented informative dialogues. In this context, we give a definition of what we call a communicational deviation. We describe the classification of the deviation occurrences showing the two categories content deviations and role deviations, each category being composed of several types. A quantitative analysis gives some details about the frequency of the deviation by type and specifies the classification by attributing a degree of importance for each type. We analyze the relation between the "cost of recovery" of a deviation and its frequency. This paper shows that deviations which appear frequently are not the most difficult to recover. The strategies used by the speakers to avoid communication failure are efficient. This study is a starting point for dialogue management systems design.

KEYWORDS: dialogue management, efficiency, communicational deviation, cost of recovery.

INTRODUCTION

In Human-Computer Dialogue (HCD) management, the real problem is to manage misunderstanding in order to increase the efficiency of the dialogue. This starting point leads us to a different approach to HCD modelling and implies taking human dialogue deviation as a basic design principle. We adopt a linguistic approach for this problem. Our method consists in analyzing real data (several human-human dialogues in a task context) and identifying when and how the dialogue strays from its aim.

We first give a brief description of the studied corpus. Then we provide a definition of deviations in finalized informative dialogue (FID) with regard to the successful communication. Next we present the deviation categories and we discuss the frequency of each type. Finally we specify the classification with the cost of recovery for each type.

1 DESCRIPTION OF THE CORPUS

The corpus we worked on was gathered for the ESPRIT 3 European project called "INTUITIVE" (INTeractive Users Interface and Tools for Information in a Visual Environment) whose objective is the design and implementation of a multimodal query system including speech as input with application to the maritime domain. The INTUITIVE maritime demonstrator is intended to be a system enabling the commanding crew of a
ship faced with a critical situation to access relevant information [1].

The corpus analyzed below consists of recorded oral dialogues between ship-masters, who will be the future users of the system, and an experimenter (sometimes two) who provides the information they need to perform a task (extinguish a fire).

We advocate a situated and empiric methodology: we study human dialogues relating to a task in a setting which was a close simulation of real system use, one of the two dialogue partners playing the role of an intelligent query system. We set up a feasible experimental protocol to gather real linguistic data [2]. At present, there is no database, all the information the master needs is on paper. We chose the situation of putting out a fire because it requires obtaining a lot of information (about the structure of the ship, the cargo, the extinguishing means, etc.).

We met masters and captains. We gathered a corpus composed of thirteen oral dialogues. The total recording lasts around eight hours.

The following table gives details about the number of speech-turns for each speaker and each dialogue.

<table>
<thead>
<tr>
<th>dialogue N°</th>
<th>number of speech-turns for each speaker</th>
<th>total number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>Q</td>
</tr>
<tr>
<td>1</td>
<td>45</td>
<td>49</td>
</tr>
<tr>
<td>2</td>
<td>67</td>
<td>66</td>
</tr>
<tr>
<td>3</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>4</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td>6</td>
<td>89</td>
<td>132</td>
</tr>
<tr>
<td>7</td>
<td>104</td>
<td>103</td>
</tr>
<tr>
<td>8</td>
<td>75</td>
<td>157</td>
</tr>
<tr>
<td>9</td>
<td>96</td>
<td>240</td>
</tr>
<tr>
<td>10</td>
<td>56</td>
<td>167</td>
</tr>
<tr>
<td>11</td>
<td>30</td>
<td>78</td>
</tr>
<tr>
<td>12</td>
<td>58</td>
<td>59</td>
</tr>
<tr>
<td>13</td>
<td>80</td>
<td>86</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Table describing the corpus (number of dialogues, of speakers and of speech-turns)

In some dialogues we have two speakers: the questioner Q (the master or captain) and the answerer A (the experimenter). In others we have introduce a second experimenter who plays the role of the crew (C).

The aim of this experiment was to obtain information requests produced by the future users of the system in order to model their way of speaking.

2 DEFINITION OF DEVIATION

In a FID, the questioner tries to obtain relevant information as quickly as possible to perform some task. The answerer helps him for this goal. So we defined the dialogue as a shared space for agreement. Its function in communication is clarification, confirmation and completeness [3].

In this context, we assume that the two basic features of successful communication are relevance and quickness of the answers. And of course the two speakers must share common knowledge about the discourse domain.

We do not use the term "error" as is frequently found in related studies [4] but we introduce the term "deviation" for two reasons. First, error refers to a normative way of studying language and particularly writing. We all know that speech is seldom regular. Structures of real spoken linguistic data are complex and involve overlapping, backward sequences and repetitions. The notion of deviation seems to be suitable to finalized dialogue because it really shows that the speakers stray from the aim of the interaction.

3 DEVIATIONS CLASSIFICATION

We analyze the corpus by studying why the answer is not direct for each information query. This approach leads us to define each case of deviation. We establish two categories of communication deviations:

- Content deviations;
- Role deviation.
The first includes Content deviations caused by information problems about the universe discourse. The other includes Role deviations caused by interaction problems particularly the speakers' functions in the dialogue.

![Deviation categories](image)

**Figure 2: Deviations classification**

The classification we propose takes into account the reasons and the consequences of the deviations.

### 3.1 CONTENT DEVIATIONS

In a FID, the interlocutors deal with application domain objects. In such a dialogue, the emphasis is put on the information exchanged. In content deviations, some unidentified references involve misunderstanding. This category is linked to knowledge representation.

The content deviation category includes five types:

- **Fuzzy question**;
- **Incomplete question**;
- **Incorrect knowledge**;
- **Unshared knowledge**;
- **Ambiguous question**.

#### 3.1.1 FUZZY QUESTION

Fuzzy question implies multiple interpretations. The interlocutor does not discern the communicational target of the question because it is imprecise. The answerer reacts with a clarification request.

| Q51 | yes yes how can nitric acid be extinguished? what / what do we / do we need regarding fighting means? |
| A51 | what do you want exactly? |
| Q52 | to know if we... if we have to use water or co2 or foam |

The "how" question is vague (Q51), it is an open question (the answerer has an unlimited choice of answers). The aim of the question is fuzzy. The utterance hesitantly pronounced probably has a consequence on the answerer's reaction (A51: clarification request). The answerer starts from the syntactical structure to build his indirect query ("if"). The questioner reformulates his query with suggestions in order to guide the answerer.

### 3.1.2 INCOMPLETE QUESTION

The interlocutor does not understand because he needs some missing information to interpret and answer the question. The answerer asks for additional information to complete the interpretation.

| Q36 | the stowage plan, yes, which locates er... |
| A37 | so you want the plan / what do you want? |
| Q37 | the stowage plan |
| A38 | of, of what? |
| Q38 | of this part |
| A39 | of deck two? |
| Q39 | of deck two |

A40: position one

Q37 is an Incomplete question. The answerer needs missing information in order to reply. He asks for more information. The questioner gives it with a pointing gesture (Q38). The answerer asks for confirmation (A39). It is surprising that the answerer needs to re-establish the reference of the phrase "this part" (anaphoric value). In Q39, the answerer confirms and the exchange achieves its goal.

### 3.1.3 AMBIGUOUS QUESTION

Ambiguous question implies multiple interpretations. The answerer hesitates between opting for one of two (or more) interpretations. Then he asks an alternative question (X or Y).
The query is ambiguous, the answerer hesitates between two interpretations. The answerer uses an alternative question to clear up the ambiguity. The deviation involves domain knowledge: there is a semantic ambiguity in the phrase "container number". It could be its identification number or its location number on the stowage plan. The answerer cannot interpret the query (Q13). The answer (A13) is a clarification request with an alternative question form in order to obtain the reference. The answerer asks for the exact signification.

3.1.4 DOMAIN KNOWLEDGE

Domain knowledge comes from the task context. We advocate that it is essential for successful communication that speakers share mutual knowledge. The introduction of new themes must be known by both speakers. When this is not the case, one of them shows his failure to understand. The repetition of the reference is made until understanding is achieved.

These deviation types just concern the lexical level: the object referred to in the request is unknown or unidentified. We distinguish two cases:

- Incorrect knowledge;
- Unshared knowledge.

In the first case, one of the speakers has a incorrect representation of the domain knowledge. The interlocutor shows his incomprehension by a negative evaluation. In the second case, one of the speakers does not know an object of the domain.

3.1.4.1 Incorrect knowledge

This deviation concerns incorrect knowledge of the discourse universe. It comes from a wrong belief. The questioner shows it but he “saves the interlocutor's face” by not pointing out that he has a wrong domain representation.

The following example shows a divergent interpretation of the container location:

- Incorrect knowledge of the container number
- Incorrect knowledge of the container position

The two speakers examine the stowage plan. This part of the dialogue is a negotiation about the container location. Both interlocutors verify that the terms they use refer to the same reference. The answer to the initial request is a clarification request about the object reference. The deviation is about the reference of the terms "first stack" and "stack". This is special knowledge of the maritime domain that both speakers do not share. The questioner clears up the ambiguity by giving his interpretation of the phrases "it must be..." (Q7), "ok, if we call it..." (Q53) and "I wouldn't call it that, I'd call it..." (Q52).

The face saving accompanies the weakening of the assertion by using of the conditional tense "it would be this one" (Q7), with the focus on the enunciation and modulation of the speaker's utterance "I wouldn't call it that, I 'd call it..." (Q52) and
with an introductory term "well, right, it's that one, ok" (Q5).

In this example the answerer shows an interpretation divergence (A4; well no), despite that the questioner tries to make the interpretations converge "yeah, ok, if we call it the stack yeah, well right it's that one ok" (Q31). After the expert explanation, the answerer agrees "uh-huh, ok" (A33).

3.1.4.2 Unshared knowledge

In general conversation, we know that speakers need to have shared background knowledge about the discourse universe [5]. In the case of unshared domain knowledge, the answerer does not know the reference of the request object. He uses a clarification request or a definition request to obtain an unknown reference.

In the following example, the answerer does not seem to know what "capacity plan" is. So the dialogue takes a long time:

Q20: "right in this case I can ask the captain to fetch the capacity plan on board, so I ask the captain" fetch the capacity plan.
A27: capacity plan?
Q21: yes
A28: what do you try to obtain in the capacity plan?
what do you try to know in the capacity plan? I try to know if the container is in is in contact with a ca/ce or capacity of er fuel that may explode if the temperature rises. <comments on the map> here are your capacities, it's on dock 2 section 9 and which number has it got, the number not the inco number but its location on the plan number?

In A27, it is a typical case of unshared knowledge. The reference is not identified by the answerer, the term "capacity plan" has not been introduced before. The answerer shows his ignorance (A27). But the questioner understands A27 as a request for confirmation.

Role deviation points out a malfunction in the identification and the distribution of the speakers' role. Role deviations concern the interaction rules that the speakers must follow [6]. We distinguish four types of role deviation:

- Initiative reversing;
- Evaluation of the relevance;
- Identification of the roles;
- Evaluation of the skill.

3.2.1 INITIATIVE REVERSING

In a FID, the initiative is usually with the questioner. It is he who manages the dialogue by imposing the discourse theme. Thus, we find cases when the answerer takes the dialogue in hand.

Initiative reversing shows the answerer manages the dialogue instead of the questioner as in the following example:

A32: position one of the stack?
Q35: ok, so can I get the extin/ extinguishing means?
A36: you asked me for the stowage plan a few minutes ago?

The answerer does not reply (Q35). He starts on another completely different subject concerning a request mentioned before (during the dialogue) that the questioner has dropped. The recall of the mentioned question is an explicit mark of the dialogue history.

In the above example, the introduction of a new theme is not reserved to the questioner as is usually the case. On the one hand, the answerer is not cooperative because he does not reply. On the other hand, he is more than cooperative because he changes the theme by making a suggestion.

In the second example, the questioner who generally gives the dialogue its dynamics, shows a certain passivity. The answerer makes some effort in order to restart the dialogue:

A73: do you want any more information?
Q162: no I'll wait
A74: shall I give you some more
Q163: er about the ship er
The answerer tries to motivate the questioner again in order to continue the interaction. So the questioner begins a new request (Q46).

3.2.2 EVALUATION OF THE RELEVANCE

The worst case of role deviation in a FID is the irrelevant question with respect to the task. The answerer uses an evaluative reaction of the relevance to return to a relevant object.

With Evaluation of the relevance, one of the two speakers shows his interlocutor that his utterance is irrelevant [7]. We consider this case to be a serious communication deviation.

The following dialogue seems to run well if we consider the series of questions and answers. But, the answerer reacts negatively to the irrelevance of the request:

Q4: well it's a roll-on-roll-off well, they're containers then?
A2: yes they are
Q5: just containers?
A3: er... we're only interested here in containers [...]

The utterance A3 is not really an answer. Q5 is a total question, Q expects a Yes or No answer. Communicational completeness is achieved because there is an answer but it does not concern the request content. The exchange Q5 - A3 is an important failure of the informative function: the answerer judges question Q5 irrelevant to perform the task and he does not answer it directly.

3.2.3 IDENTIFICATION OF THE ROLES

In a FID which is a codified dialogue, it is important that the interlocutors know what they have to do and what they can do. When doubt arises about "who does what" and "who knows what", this implies a divergence, an Identification of the roles deviation.

In this type, the interlocutors' interaction roles have not been clearly identified. In this floating roles case, the questioner does not know what the answerer knows exactly or the answerer does not know what the questioner expects of him. They both have to clarify their interaction functions.

In the following example the answerer does not understand what the questioner expects of him:

Q56: so we can see from the computer or what the... the containers contain the ones there? if we know that?
A56: do you need any information or documents?

The answerer is asking "what do you expect of me?".

3.2.4 EVALUATION OF THE SKILL

The questioner asks the answerer for Evaluation of the skill but he cannot give it. This generates a deviation. The interactional situation is a simulation of a query system but it is not a support tool. In fact, the answerer cannot judge or evaluate. His role is clearly defined; he only gives information requested by his interlocutor in order to perform his task.

Q45: get me the dangerous cargo office, question, if I attack if I neutralize the deck, do I have s / first have I got any neutralize equipment of the goods hold?
A41: what do you mean?
Q46: have I got a means for example pressurized?
A42: yes
Q47: well the dangerous cargo office will you tell me er... can I use / is there any chance can I use co2 for this type of fire?
A43: [silence] well the dangerous cargo office will put you in touch with, will fax you a declaration from Merck/
Q48: [...] can I use co2 or not?
A44: yes you can use co2

The reference of the phrase "inert equipment" having been re-established, the dialogue can continue. Thus, it seems that the expected answer (by the request Q47) is a yes or no reply. The given answer (A43) is not satisfactory. The utterance A43 does not match the questioner's expectation. Q47 is an Evaluation of the skill request "is there any chance... can I use co2 for this type of fire?". The questioner trusts his interlocutor, he considers him to be a domain expert. This is a "negotiation" of the answerer's role.

The questioner does not understand what the answerer suggests, which is why he repeats his request with some irritation ("can I use co2 or not?"). The dialogue sinks deeper
into difficulties. The questioner loses patience (Q48) and uses an alternative question (yes or no) in order to make his interlocutor reply as he expects.

This is a complex sequence because it also presents a content deviation. The questioner does not seem to know what the answerer is speaking about in A43; the phrase "declaration from Merk" means nothing to him. The answerer obviously believes that his interlocutor knows what he is speaking about; the answerer thinks that the questioner knows the mentioned Merk chemical company which can give information about the loaded chemical products and their safety. This can be also a deviation connected with the roles distribution because the answerer believes that he has to give some documents.

Remark: From a task point of view, it is strange that the questioner asks if he can use co2 before asking if there is any co2 equipment. This is a problem related to the task scheduling.

4 DEVIATION FREQUENCY

The total number of deviations identified in the corpus is 219. The below figure presents the proportional distribution of the deviation in the corpus according their category content deviations or role deviations.

![Figure 3: Proportional distribution of deviations by category](image)

We notice that two thirds of the deviations used in the corpus are related to the rules of the interaction the speakers have to follow.

The following figure shows more precisely the distribution of deviations by type.

![Figure 4: Distribution of the deviations by type in the total corpus](image)

We notice that the four most frequent types are:

- Initiative reversing (54 occurrences);
- Evaluation of the relevance (46 occurrences);
- Fuzzy question (37 occurrences);
- Identification of the roles (36 occurrences).

The total of these four types is 173 (out of the 219 of the corpus). The greater part of these deviations are role deviations related to the interaction rules except the Fuzzy question which is a content deviation.

The other five types are under-represented.

- Incorrect knowledge (16 occurrences);
- Incomplete question (11 occurrences);
- Unshared knowledge (8 occurrences);
- Ambiguous question (8 occurrences);
- Evaluation of the skill (3 occurrences).
5 RECOVERY COST OF DEVIATION

To describe a deviation completely, we need to know if it is important, that is to say, if it takes a long time to recover. So we study how long it takes the speakers to bring the dialogue back to its purpose after a communicational break-down. For each deviation listed in the corpus (219 occurrences) and by type, we count the number of speech-turns involved in a deviation recovery.

After recovery, we consider that the dialogue is back on the right track when the questioner asks a new query making the theme progress.

Among the 2753 speech-turns of the total corpus\(^1\) 1275 are used in deviations and their recovery.

The following figure shows the distribution of the type deviations by the number of speech-turns involved in their recovery.

![Graph showing the distribution of type deviations by number of speech-turns](image)

The number of speech-turns needs to be specified. In the following diagram we give the maximum, minimum and mean numbers of speech-turns found in the corpus for each type.

![Graph showing maximum, minimum, and mean numbers of speech-turns](image)

Figure 6: Maximum, minimum and mean numbers of speech-turns involved in deviation recovery

For instance, the description of the type Fuzzy question can be read in this way:

- the greatest number of speech-turns involved in the recovery of this deviation is 19;
- the smallest number of speech-turns involved in the recovery of this deviation is 3;
- The mean number of speech-turns involved in the recovery of this deviation is 6.81.

The following classification gives an increasing order of the deviations according the cost of recovery (mean number of speech-turns).

- Ambiguous question (4.50);
- Initiative reversing (4.89);
- Evaluation of the relevance (5.20);
- Identification of the roles (6.42);
- Incorrect knowledge (6.44);
- Unshared knowledge (6.75);
- Fuzzy question (6.81);
- Incomplete question (7.18);
- Evaluation of the skill (8.67).

---

1 We notice that 46\% of the total turns-talking of the corpus are used in deviations recovery.
The number of speech-turns necessary to restore the dialogue after a deviation occurrence allows a more precise characterization of the classification. We consider that a deviation is more or less important or costly depending on the number of speech-turns used to retrieve it.

6 RELATION BETWEEN FREQUENCY AND RECOVERY COST

The following table shows the relation between the frequency and the recovery cost of deviations. We give the total number of deviations by type, the total number of speech-turns involved in the recovery and the mean number of speech-turns. The deviations are given in a decreasing order according to their frequency.

<table>
<thead>
<tr>
<th>Deviation Type</th>
<th>Deviation Total Number</th>
<th>Speech-Turns Total Number</th>
<th>Speech-Turns Mean Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiative reversing</td>
<td>54</td>
<td>264</td>
<td>4.89</td>
</tr>
<tr>
<td>Evaluation of the relevance</td>
<td>46</td>
<td>239</td>
<td>5.20</td>
</tr>
<tr>
<td>Fuzzy question</td>
<td>37</td>
<td>252</td>
<td>6.81</td>
</tr>
<tr>
<td>Identification of the roles</td>
<td>36</td>
<td>231</td>
<td>6.42</td>
</tr>
<tr>
<td>Incorrect knowledge</td>
<td>16</td>
<td>103</td>
<td>6.44</td>
</tr>
<tr>
<td>Incomplete question</td>
<td>11</td>
<td>79</td>
<td>7.18</td>
</tr>
<tr>
<td>Ambiguous question</td>
<td>8</td>
<td>36</td>
<td>4.50</td>
</tr>
<tr>
<td>Unshared knowledge</td>
<td>8</td>
<td>54</td>
<td>6.75</td>
</tr>
<tr>
<td>Evaluation of the skill</td>
<td>3</td>
<td>26</td>
<td>8.67</td>
</tr>
</tbody>
</table>

Figure 7: Table summing up the frequency and the cost of recovery for each deviation type.

We want to see if there is a correlation between the frequency and the recovery cost of the deviations. Let’s begin with the most frequent deviations identified in the corpus:

- The Initiative reversing type is the most frequent deviation (54 occurrences out of the 219 of the corpus). But this kind of deviation is not the most costly (4.89 mean speech-turns).

- The Evaluation of the relevance type is also frequent (46 occurrences out of 219) and is not costly too (5.2 mean turns).

- This remark is not so obvious for the frequent types Fuzzy question (37 occurrences) and Identification of the roles (36 occurrences) whose means are fairly low but not the lowest (6.81 and 6.42 mean turns).

On the other hand, let’s examine the cost of recovery for the less frequent deviations:

- The Evaluation of the skill type is the least represented (3 occurrences out of 219). However this deviation is the most costly (8.67 mean turns).

- The types Unshared knowledge (8 occurrences) and Incomplete question (11 occurrences) are little represented and these deviations are costly too (6.75 and 7.18 mean turns).

- It is more difficult to conclude for Ambiguous question type. It appears as rarely as Unshared knowledge (8 occurrences) but it has a lower cost of recovery (4.5 for 6.75 mean turns).

We can explain this exception by the nature of the deviation and the kind of strategy used for recovery. The deviation study has shown that the Ambiguous question type is generally easy to retrieve with an alternative question. This strategy of recovery is not very complex which is why it takes few turns to recover Ambiguous question deviation.

If we consider the results of the table above (cf. Figure 7) from a general point of view, we can notice that the more the deviation is costly the less it appears.

The relation between the frequency and the cost of recovery leads us to conclude that the costlier the deviation is, the less frequently it appears. This general tendency is justified because our study has shown that the speakers in a FID use strategies in order to make the interaction successful. So it is logical that the worst case seldom happens.
CONCLUSION

The speakers in a query task-oriented dialogue try to manage the communication efficiently in order to get the required information and to act. So they use discourse strategies to avoid communication failure or to recover deviations when they occur [8].

We propose a study of human communicational deviation with a view to computerization. It is quite impossible to integrate all the strategies used in human dialogues to avoid failure because there are so many. But, we can try to identify the deviations according their types and their cost of recovery. We now know that some deviations are easy to retrieve and others involve more effort and more speech-turns.

User-friendly dialogue systems have to adapt their way of communicating to human communication deviation. In Human-Computer Dialogue Management, the system must be able to identify a deviation. First, we only need to know which type of deviation it is and it will be easier to recover it. So the issue is to model deviations with a view to recovering them. The model has to suggest solutions to manage deviations. The dialogue management module has to foresee more than one strategy to carry out the dialogue. This study proposes certain recommendations concerning the efficient communication [9].

BIBLIOGRAPHY


SPEECH ACT GENERATION IN COOPERATIVE DIALOGUE

Robbert-Jan Beun
Institute for Perception Research (IPO)
P.O. Box 513
5600 MB Eindhoven
The Netherlands
e-mail: rjbeen@matlab.research.philips.com

ABSTRACT

This paper presents a simple model for speech act generation in cooperative dialogue. Main goal of the dialogue is the collaborative exchange of information so that, if the information is available, in the final state the participants believe the answer to an initial question. The participants' communicative strategy is determined by the rules of the dialogue game. On the one hand, the rules regulate which speech act will be performed depending on a certain information state of one of the participants, and, on the other hand, they dictate how an information state will change as a result of the performance of a particular speech act. The resulting structure of the information exchange can be complex, since the knowledge to find an answer to the initial question may be distributed among the participants and may therefore result in subdialogues and the generation of counter-questions.

INTRODUCTION

In his book on Pragmatics, Levinson argues that, in contrast with more theoretical oriented approaches towards conversation, the proper way to study conversational organization is through empirical techniques (Levinson, 1983, Chapter 6). Although I do agree with Levinson that an empirical approach in the study of human communicative behaviour is indispensable and that so far empirical approaches have made important contributions in the area of a systematic description of conversational phenomena, it is my opinion that a radical empirical approach, without careful modelling of the basic concepts and the theoretical principles of conversation, would be like practising physics without mathematics.

In this paper, I will therefore propose a formal system that, although simple in its basic form, enables us to produce abstract conversations of which some of the properties agree strikingly with some properties found in conversation by Conversation Analysts. In line with Levinson, I will take distance from the idea that conversation can be modelled by a concatenation of speech acts regulated by a set of sequencing rules or grammar. Modelling the properties of conversation demands a notion of context, or at least two contexts, one for every participant, and the coherence of the speech acts is merely a result of the conversational rules that operate on these contexts (see also Searle, 1992; Dascal, 1992).

Central in the approach is that participants in a dialogue generate speech acts to transfer relevant information with respect to their goals and that their information state changes as a result of the interpretation of these speech acts (see e.g. Gazdar, 1983 and Bunt, 1990). The basic goal in the dialogue is defined by an initial question asked by one of the participants that has to be answered in a cooperative manner. Like in realistic dialogue situations, I will assume that information is distributed among the participants and that, as a consequence of the distributed information, the dialogues may be complex, and may result in subdialogues and the generation of counter-questions.

BASIC ISSUES IN DIALOGUE

The need for communication in a dialogue situation seems to come from a fundamental property of human cognition: participants in a dialogue have partial information about the domain of discourse and their partner, and the information
is distributed among different information states which are only (indirectly) accessible by sending messages from one agent to the other. The intended message has to pass through an error-prone communication channel, causing the sender to be uncertain whether the transmitted message will arrive correctly. Also, the message has to be encoded for transfer. A speaker has to select relevant information, to order the information, to formulate the message into words and grammatical structure, and finally, to articulate the message to produce overt speech (Levelt, 1989). In machine communication the decoding process is an exact inverse of the encoding; in human communication, however, the process of encoding and decoding are never (inverse) identical, so important mismatches may occur between the generation and interpretation of the intended message. For instance, the speaker may use words that are not part of the interpreter’s lexicon, may produce sentences that are ambiguous with respect to the interpreter’s information state and current goals, or may assume presuppositions that conflict the interpreter’s information state.¹

These mismatches often give rise to so-called subdialogues, i.e. insertion sequences included in the dialogue to indicate that the receiver has an interpretation problem and, for instance, to elicit extra information from the dialogue partner (e.g. Schegloff, 1972; Dik, Dubber & Weijdemo, 1978; Allwood, Nivre & Ahlsén, 1992). In order to avoid these complex repair problems, a speaker sometimes starts with a pre-sequence, for instance, to introduce the relevant concepts or to establish common knowledge about the presuppositions in the main speech act (Levinson, 1983). Hence, these mismatches or the avoidance of mismatches give a dialogue participant a reason to generate a particular language fragment other than the main message.

With some exceptions (e.g., Appelt, 1985; McCoy, 1989; Moore & Swartout, 1991), the work on computational models of natural language generation has focused on lexical choice and text planning, often based on Rhetorical Structure Theory (Mann & Thompson, 1988), given a specific intention or goal of the language producer (see, for instance, McDonald & Bole, 1988; Dale Mellish & Zock, 1990; Paris, Swartout & Mann, 1991).

¹This problem is not exclusive for natural language interaction; other modalities, such as graphics and sound, may be subject to the same deficits. Visual or auditory metaphors and symbols may be chosen inappropriately with respect to the receiver’s information state or may be poorly linked to an existing linguistic discourse.

But preceding to the encoding and decoding process there is yet another even more fundamental problem, i.e. what type of speech act and content should the speaker transfer in order to establish adequate communication given the current situation and the goals of the dialogue participants? An answer to this question requires at least a better understanding of what we mean by the current situation in a dialogue.

THE TRIANGLE METAPHOR

The discussion below will be restricted to dialogues performed in a situation where people try to solve specific tasks in a cooperative and rational manner, i.e. dialogues with information desks, in instruction, or even with a computer system. The view on communication is based on a simple model employed in human-computer interaction (Hutchins, 1989; Ahn et al., 1995). Underlying this model is the recognition that humans interact naturally with their environment in two ways: symbolically and physically. On the one hand, if there is an intermediary interpreter, humans can interact symbolically and use language to give commands, ask for or provide information, etc. On the other hand, physically, one picks up objects, moves or fastens them, or observes them by seeing, feeling, hearing or smelling. The essential difference between the two types of interaction is that actions of the first type (for instance, speech acts; Austin, 1962; Searle, 1969) need an interpreter who can bridge the gap between the symbols and their actual meaning and purpose, while actions of the second type are related in a more direct manner to human perception and action.

In parallel with the distinction symbolically vs. physically, humans engaged in these types of dialogue can perform two types of external actions: a. communicative actions in the sense of Grice’s non-natural meaning (Grice, 1987), i.e. intended to cause some cognitive effect in the recipient, and b. non-communicative actions to observe or change particular properties of the domain. It goes without saying that the two types of actions can be considerably interrelated (Grosz & Sidner, 1987; Airenti, Bara & Colombetti, 1993). In addition, we will distinguish a second type of non-communicative action that is carried out towards the internal information state (e.g. beliefs and desires) of the participant him- or herself, namely updating. Updating is the process of adjusting the information state as a result of observation,
reasoning or goal-adaptation.

The distinctive communication channels are represented in the so-called triangle metaphor (Figure 1.), where the angles represent the domain of discourse and the two dialogue participants; the arrows between the angles represent the flow of information between the participants and the domain. We will say that information flows from $X$ to $Y$ – where $X$ and $Y$ range over both participants and the domain – if the state of $Y$ changes as a result of the state of $X$ (cf. Dretske, 1981). If $X$ represents the domain of discourse, the information not necessarily flows as a result of a change of the state of $X$. For instance, an observation of a static domain (e.g., a picture) changes the information state of the observer, but no changes will take place in such a domain. The external actions can now easily be expressed in terms of the flow of information between the angles of the triangle. A communicative act performed by $X$ towards $Y$ is a flow of information from $X$ to $Y$; a domain observation is a flow of information from the domain towards the observer and an action carried out in the domain is a flow of information from the actor to the domain.$^3$

**Figure 1:** The triangle metaphor.

MODELLING

COOPERATIVE BEHAVIOUR IN

A DIALOGUE SITUATION

In the type of dialogue that we consider, we will make some important simplifications with respect to the model presented in the previous section. We will assume, for instance, that both participants have no access to the domain of discourse and, therefore, are unable to change or observe the domain. In other words, there is only a flow of information between the two participants. Moreover, only two types of information about the domain of discourse will be distinguished: a. simple propositions ($p, q, r, \ldots$) and b. compound propositions consisting of a simple proposition, an arrow and a simple proposition ($p \rightarrow q, \ldots$).

The communicative acts

The dialogue will be considered as a set of moves in a game, the communicative acts (see Carlson 1983). Each communicative act has a function (or goal) and (propositional) content, and changes the information state of the participants. Participants may use four types of communicative acts: questions ($p?X$), statements ($p!X$), an indication that certain knowledge is not available ($p?!X$) and a closure of the dialogue ($\emptyset$). Questions are answered if the information is available. This may take more than one turn, because the knowledge to answer the question may be distributed among the participants. In accordance with the Gricean maxims of cooperation (Grice, 1975), both agents are forbidden to ask anything that they believe to be true or that they believe their partner does not believe. Also, they are not allowed to state anything they do not believe to be true. At the end of the dialogue, all questions should be answered or a reason should be given why a particular question could not be answered.

The communication channel

In practice, the communication channel between the participants may cause messages to be delayed (as in letters) or disturbed; the channel can be duplex, half duplex, etc. In this paper, however, the communication channel between the partners is ideal, i.e. no propositional information gets lost if transferred between the participants. Only one participant can talk at a time, so the channel is half duplex. Time is unimportant, but the order of communicative acts is important, since they change the belief states of the agents.

The agents' beliefs

Two types of belief will be distinguished:

* private beliefs of an agent about the domain of discourse ($B_X b$)
• beliefs of an agent about what the partner does not believe \( (B_x \neg B_y p) \)

Information in the belief state can be assimilated in two ways: 1. the existing belief state of the agent him- or herself and 2. communicative acts produced by the dialogue partner. The first case refers to conclusions drawn from already existing beliefs; here, a simple reasoning mechanism is included. The second case refers to the possible extensions of the belief state as a result of communicative actions.

In this paper, the participants have no special expertise. They simply consider their belief to be true and give priority to their own beliefs, whatever the topic of a declarative utterance by their partner. The content of a declarative is simply added to the belief state of the hearer if he or she has no information about it. Beliefs about the domain of discourse can be incorrect, however, but the dialogue partners will never know this, since they have no access to a domain of discourse. Only by asking questions to their partner, the knowledge may become available.

The agents' intentions

Usually participants in a dialogue have different goals and intentions. For instance, an informant at an inquiry desk probably wants to earn money for a living and the information seeker probably wants to be at a certain place at a certain time. In some cases, it is not at all in the interest of one of the participants to provide information (for example, in a police interrogation).

In this paper, the participants only have communicative intentions, since they have no access to the domain of discourse. One of the participants ('the mistress', M) initiates the goal of the dialogue and starts with a question (the origin of the question falls outside the scope of this paper), while the other ('the servant', S)\(^4\) follows the mistress and reacts to her wishes in a cooperative manner. Hence, the mistress gives relevance to all the actions that are performed in the dialogue. As long as we have not indicated otherwise, both the mistress and the servant act according to the same dialogue rules, but usually they have discrepancies in their respective information states.

Communicative intentions will be notated as follows: \( B_x W K_Y (\ldots, p) \), meaning that 'X believes that Y wants to know whether p'. In-

\(^4\)In this paper, the servant will receive masculine pronounization.

\( W_K Y (\ldots, p) \) means that the dialogue partner Y has the communicative intention to know the truth value of several different propositions, of which \( p \) is the last one. \( W_K Y (\ldots, p, \ldots) \) means that \( p \) is part of the list of communicative intentions, but not necessarily the last one. The list may be empty, indicated by \( W_K Y(\emptyset) \), meaning that Y has no communicative intentions.

The update of information states

The update function yields a new information state depending on the old state and the just performed communicative act. Vice versa, the new information state of a dialogue partner contains the preconditions for the next move. In other words, a move is completely determined by the information state of one of the participants at a certain time and the rules for cooperative behaviour that will be presented below. This implies that the next move is not only based on the previous one, as would be the case with a dialogue grammar based on communicative acts, but also on the previous information state of the partner.

We will assume that, if a participant believes that his or her partner wants to know whether \( p \), then he or she also believes that the partner does not believe that \( p \) (R1). Moreover, we assume 'modal Modus Ponens', i.e. if agent X believes that \( p \) and he or she believes that \( p \rightarrow q \), then he or she believes that \( q \) (R2).

\[ R1. \quad B_x W K_Y (\ldots, p, \ldots) \rightarrow B_x \neg B_y p \]
\[ R2. \quad B_x p \& B_x (p \rightarrow q) \rightarrow B_x q \]

The private belief state is monotonic, i.e. everything that can be inferred from previous private belief states, can also be inferred from new private belief states. Information about the intention state \( B_x W K_Y \) and about the non-beliefs of the other \( B_x \neg B_y \) can be changed after particular communicative acts. For example, if a question 'whether \( p \) has been answered, the intention to answer this question will be dropped. We will not be concerned with the details of the update mechanism, but assume that the information state will updated according to the principles R1 and R2.

To represent the consequences (or postconditions) of a particular communicative act, the twisted arrow '\( \rightsquigarrow \) is introduced. Left side of the arrow is of type action, the right side is of type proposition. On the left side of the arrow we introduce the just performed communicative act of a participant, on the right side the new information state as a result of that communicative act.
Below, \( \phi \) means that, if \( p \) exists in a particular belief or intention list, \( p \) will be removed from that particular list.

In update rule U1 it is expressed that if \( X \) utters a statement with content \( p \), afterwards the recipient, \( Y \), believes that \( p \) and \( X \) does not believe anymore that \( Y \) does not believe that \( p \).\(^5\) If \( X \) utters a question with content \( p \), afterwards \( Y \) believes that \( p \) is a communicative intention of \( X \) and \( p \) is added at the end of the intention list (U2). If \( X \) indicates that he does not know that \( q \), afterwards \( Y \) believes that \( X \) does not believe that \( p \) and \( p \) is removed from \( X \)'s intention list of \( Y \) (U3). In U4 it is expressed that the information states do not change after a closing of the dialogue.

**Update**

1. \( p_X \sim B_Y p \& B_X \neg B_Y (\phi) \)
2. \( p_X \sim B_Y W K_X (\ldots , p) \)
3. \( p_X \sim B_Y \neg B_X p \& B_X W K_Y (\ldots \phi) \)
4. \( \$X \sim \emptyset \)

**The rules of the dialogue**

The participants' communicative strategy is determined by the rules of the dialogue game which determine the communicative act that will be performed depending on a certain information state of one of the participants. An explicit link between the preconditions of the act and the act itself is established by the double arrow `\( \Rightarrow \)`. Left side of the arrow is of type proposition, the right side is of type action. On the left side we introduce the preconditions in terms of the information state of the participant, on the right side the generated communicative act as a result of the precondition.

In generation rule G1 it is expressed that if \( q \) is the last item on the intention list of participant \( Y \) and \( q \) is believed by \( X \), then \( q \) will be answered by \( X \).\(^6\) If \( X \) does not believe \( q \), however, but believes that there is a solution to solve \( q \) by knowing the answer to \( p \), then \( X \) will ask for the truth of \( p \) (G2). If \( X \) does not even know a solution to solve \( p \), then \( X \) has to admit that he or she does not know the answer (G3). Finally, a closure communicative act is generated if the intention list is empty (G4).

**Generation**

\(^5\)Note that, in correspondence with rule R1, \( p \) will also be removed from \( B_X W K_Y (\ldots , p) \).

\(^6\)In fact, we should add in all cases that it is \( X \)'s turn to speak. We have left this out for reasons of legibility.

\[ G1. B_X W K_Y (\ldots , q) \& B_X q \Rightarrow q ! X \]
\[ G2. B_X W K_Y (\ldots , q) \& \neg B_X q \& B_X (p \rightarrow q) \& \neg B_X \neg B_Y p \Rightarrow p ! X \]
\[ G3. B_X W K_Y (\ldots , q) \& \neg B_X q \& \neg B_X (p \rightarrow q) \Rightarrow q ! X \]
\[ G4. B_X W K_Y (\emptyset) \Rightarrow \$ X \]

Closing of the dialogue should only give the turn to the next speaker. To avoid an infinite sequence of closing communicative acts, a meta-rule has been defined to close the dialogue:

**Closing**

Both dialogue partners stop generating communicative acts iff two successive closing communicative acts are performed (i.e. the sequence \( \$ X \& \$ Y \)).

**AN EXAMPLE DIALOGUE**

Let us turn to an example where we have communication between the mistress M and the servant S (Table 1). In Table 1, we have indicated the respective information states of M and S and the communicative acts (MOVE) that occur as a result of the dialogue rules. The information that represent the preconditions for the next move are indicated in bold.

In the initial situation, S believes that \( p \rightarrow q \) and believes that \( r \rightarrow q \). M believes that \( s \rightarrow p \) and believes \( r \). M starts with the initial question \( q ! R \). S does not know the answer to the question directly. However, if S comes to know either \( p \) or \( r \), then S can solve the problem by rule R2. So, according to rule G3, S has to generate the question whether \( p \) or the question whether \( r \). In Table 1, S starts with the first one. M can solve this problem by asking for \( s \), but S has to inform M that he has no information about \( s \). S was on the wrong track and the goal s has been dropped (rule U3). The goal \( q \) is still not solved and \( p \) cannot be asked for, since S believes that M does not know anything about \( p \). Now, \( r \) can be asked by S, and, since M has direct information about \( r \), M can answer the question. In turn, S can answer M's original question and finally, the dialogue can be closed since all communicative intentions are removed.

Below, the same example has been presented in natural language. The propositions that were used in Table 1 have the following meaning:

\[ p: \text{Olga smokes cigars} \]
\[ q: \text{Olga is happy} \]

\[ G1. B_X W K_Y (\ldots , q) \& B_X q \Rightarrow q ! X \]
\[ G2. B_X W K_Y (\ldots , q) \& \neg B_X q \& B_X (p \rightarrow q) \& \neg B_X \neg B_Y p \Rightarrow p ! X \]
\[ G3. B_X W K_Y (\ldots , q) \& \neg B_X q \& \neg B_X (p \rightarrow q) \Rightarrow q ! X \]
\[ G4. B_X W K_Y (\emptyset) \Rightarrow \$ X \]
<table>
<thead>
<tr>
<th>$B_S$</th>
<th>$B_S W K_M$</th>
<th>$B_S - B_M$</th>
<th>MOVE</th>
<th>$B_M$</th>
<th>$B_M W K_S$</th>
<th>$B_M - B_S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p \rightarrow q$</td>
<td>$r \rightarrow q$</td>
<td></td>
<td></td>
<td>$s \rightarrow p$</td>
<td>$r$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\text{q?}_M$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td>$r \rightarrow q$</td>
<td>$&lt; q &gt;$</td>
<td>$q$</td>
<td>$s \rightarrow p$</td>
<td>$r$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\text{p?}_S$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td>$r \rightarrow q$</td>
<td>$&lt; q &gt;$</td>
<td>$q$</td>
<td>$s \rightarrow p$</td>
<td>$&lt; p &gt;$</td>
<td>$p$</td>
</tr>
<tr>
<td>3</td>
<td>$\text{s?}_M$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td>$r \rightarrow q$</td>
<td>$&lt; q, s &gt;$</td>
<td>$q, s$</td>
<td>$s \rightarrow p$</td>
<td>$&lt; p &gt;$</td>
<td>$p$</td>
</tr>
<tr>
<td>4</td>
<td>$\text{s?}_S$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td>$r \rightarrow q$</td>
<td>$&lt; q, # &gt;$</td>
<td>$q, s$</td>
<td>$s \rightarrow p$</td>
<td>$&lt; p &gt;$</td>
<td>$p, s$</td>
</tr>
<tr>
<td>5</td>
<td>$\text{p?}_M$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td>$r \rightarrow q$</td>
<td>$&lt; q &gt;$</td>
<td>$q, s, p$</td>
<td>$s \rightarrow p$</td>
<td>$&lt; # &gt;$</td>
<td>$p, s$</td>
</tr>
<tr>
<td>6</td>
<td>$\text{r?}_S$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td>$r \rightarrow q$</td>
<td>$&lt; q &gt;$</td>
<td>$q, s, p$</td>
<td>$s \rightarrow p$</td>
<td>$&lt; r &gt;$</td>
<td>$p, s, r$</td>
</tr>
<tr>
<td>7</td>
<td>$\text{r?}_M$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q, r$</td>
<td>$r \rightarrow q, q$</td>
<td>$&lt; q &gt;$</td>
<td>$q, s, p$</td>
<td>$s \rightarrow p$</td>
<td>$&lt; # &gt;$</td>
<td>$p, s, #$</td>
</tr>
<tr>
<td>8</td>
<td>$\text{q?}_S$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q, r$</td>
<td>$r \rightarrow q, q$</td>
<td>$&lt; # &gt;$</td>
<td>$q, s, p$</td>
<td>$s \rightarrow p$</td>
<td>$r, q$</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>$\text{q?}_S$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p \rightarrow q, r$</td>
<td>$r \rightarrow q, q$</td>
<td>$s, p$</td>
<td>$s \rightarrow p$</td>
<td>$r, q$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>$\text{q?}_S$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: $S$ and $M$ try to solve the problem whether $q$ is true. Initially, the information about $q$ is distributed; in the final state, both $S$ and $M$ believe that $q$ is true.
r: ‘Olga works at IPO’
s: ‘Olga is a manager’

In correspondence with Table 1, the servant John initially believes that ‘if Olga smokes cigars, then she is happy’ and believes that ‘if Olga works at IPO, then she is happy’. The mistress Mary initially believes that ‘if Olga is a manager, then she smokes cigars’ and believes that ‘Olga works at IPO’. For some reason, Mary is interested to know the answer to the question whether Olga is happy.

In order to obtain a more natural dialogue, we have added utterances that are not generated by the rules that were defined in the model, such as opening of the dialogue, indirect questioning, thanks and particles on the level of processing (‘Aha’, ‘Well’, ‘Eh’). The basic structure, however, is isomorphic to the structure presented in Table 1.

1. Mary: Hello John, I have a question. Can you tell me whether Olga is happy?
2. John: Well, let me see, does she smoke cigars?
3. Mary: Eh... That depends, is she a manager?
4. John: Sorry, I don’t know.
5. Mary: Then I don’t know whether she smokes cigars.
6. John: Aha, wait... does she work at IPO?
7. Mary: Yes, she does.
8. John: Well in that case, don’t worry, she’s happy.
9. Mary: Great, thanks a lot, bye.

DISCUSSION

Clearly, the dialogue presented in the previous section is far from natural and many questions remain. The approach introduced in this paper, however, has an important advantage compared to the dialogue grammar approach.

In both approaches, utterance generation is based on the goal-directed behaviour of individuals in conversational settings. In the dialogue grammar approach, sequences of utterances are generated from the concatenation rules for speech acts, with the focus on the functional or illocutionary part of the speech act (such as question, statement, promise, etc.). But, as the following example suggests, depending on the content of the utterance the same sequence of illocutions can be ill-formed (A1, B1) or well-formed (A2, B2):

A1: Mary is in the kitchen.
B1: I am sorry, thanks.

A2: You are in front of the overhead projector.
B2: I am sorry, thanks.

In both cases the first utterance is a statement and the second an excuse and an expression of gratitude. In order to explain the incoherence of the first sequence, a representation of context information is crucial, especially in terms of the overall beliefs and intentions of the agents, both communicative and non-communicative. In our approach, therefore, utterances are not only generated on the basis of functional properties of the utterance, but also on the basis of the content of the utterance and the previous information state of the participants.

Also, our model does not suffer from the same computational complexity that arises in context-change approaches (see e.g. Perrault, 1990 and Beun, 1994), where there is no direct connection between the previous speech act and the next one. As a result of the closure of the axioms, in the context-change approach an infinite number of information states is modelled. In our model, only those information states are modelled that are needed to establish the appropriate communication (6 in total, 3 per participant).

We have to take into account, however, that the types of information states have to be extended in the future. It is our experience from natural dialogue experiments that deep nestings of belief (> 2), such as ‘A believes that B believes that A believes that B’, are unnecessary to model, since they do not play an important role in dialogue modelling. An important extension, however, is the inclusion of so-called ’mutual belief’ (see e.g. Clark & Marshall, 1981). First, modelling mutual belief enables the dialogue partners to leave out superfluous information and, second, it enables dialogue participants to communicate information that can be used in reference to objects.

Furthermore, our model describes (or generates) elementary dialogue phenomena which are also observed in Conversational Analysis (see e.g. Levinson, 1983). First of all, in Table 1 we can observe the local management of adjacency pairs, i.e., the prototypical regularity in the sequences of communicative acts, such as question/answer...
(moves 3–47 and moves 6–7) and the closing of the dialogue (moves 9–10). Second, we observe insertion sequences, such as move 3–4 and move 2–7. Depending on the initial question and the initial belief state, the dialogue rules generate an arbitrary number of levels of sequences and the final answer may be removed many utterances away from the initial question. Empirical research should disclose how deep human participants can go in real conversation.

An important drawback of the natural language example is that the dialogue looks still superficial, especially without the extra control utterances. An extension, along the line of Levinson's preference organization model (Levinson, 1983), in which linguistic markers are added in the dialogue if non-preferred speech act sequences are generated, seems therefore inevitable. Another interesting candidate is Bego's context change approach (Bego, 1995), where specific types of feedback are considered as side-effects of updates on the information states.

Also, the representation of domain knowledge should be extended drastically. However, extending the model with, for example, predicate logic has important consequences for the dialogue protocol, since formulas may be undecidable and, as a result, dialogue partners could speak until infinity.

An extension of the domain language may also influence the need for other types of information states. For example, the inclusion of negation introduces the possibility of conflicting or inconsistent beliefs. Clearly, we cannot simply add a proposition p to a belief state of one of the participants if he or she already believes that ¬p. In those cases, a solution could be to extend the model with belief about the partners belief, so that inconsistent beliefs can be separated. Depending on, for example, the sequel of the dialogue, the partners' expertise and the roles that the partner play in the interaction (such as teacher/student, student/student, expert/apprentice), the agent may drop his or her original belief and accept the other's belief.

This paper has been concerned with certain basic aspects of communication. To clarify fundamental principles in dialogue we have chosen a simple model based on the information states of the participants, the communicative acts they perform to exchange information, and the underlying rules for the generation of the communicative acts and the update of the information states. Exploring the model with different information states and different communicative strategies may give important insight in both human/human and human/machine communication. It should be stressed, however, that the rules for dialogue are rudimentary and that extensions along the lines indicated above will have far reaching consequences from a formal and a computational point of view.

ACKNOWLEDGEMENTS

This research was supported by the Cooperation Center Tilburg and Eindhoven Universities, The Netherlands.

REFERENCES


7Strictly speaking, move 4 is not an answer, but a denial of the precondition to answer the question.
Amsterdam, Instituut voor Algemene Taalwetenschap, report nr. 21.
PRAGMATIC INTERPRETATION AND DIALOGUE MANAGEMENT
IN SPOKEN-LANGUAGE SYSTEMS

Gert Veldhuijzen van Zanten
Institute for Perception Research
P.O.Box 513
5600 MB Eindhoven
The Netherlands
e-mail: veldhv2@natlab.research.philips.com

ABSTRACT
This paper describes the design of pragmatic-interpretation and dialogue-management modules in an
automatic inquiry system that can be consulted through spoken natural language over the telephone.
The system is designed around a central multi-level data structure representing the discourse that has un-
folded during the dialogue. At the highest level of this discourse representation the information exchange is
represented as a series of changes or updates of an information state. Conditions on the information state
give rise to actions of the dialogue manager. The dialogue manager aims at achieving the user's goal in a
manner that is understandable to the user, efficient and correct. This is no trivial problem, because nat-
ural language and, in particular, speech understanding leads to many uncertainties. To deal with uncertain
information, we have designed feedback and verification mechanisms and means for contextual under-
standing, underspecification and pragmatic inferencing.

INTRODUCTION
The understanding of natural, and in particular, spoken language involves many issues that are only par-
tially understood and thus hard to implement on computer systems. Some of these issues are ambiguity,
underspecification, contextual interpretation and pragmatic inferences. Within the NWO-TST1 priority
programme on Language and Speech Technology, we are building a number of demonstrator systems,
OVIS1, OVIS2 and OVIS32, that engage in spoken-language dialogue about train time-table inquiries. In
this paper, we focus on the design of the pragmatic-

1. NWO is the Netherlands Organization for
Scientific Research, and TST is a Dutch ac-
ronym for Language and Speech Technolo-
gy

interpretation and dialogue-management modules for
such systems. Most examples are derived from the
design of OVIS2, but the ideas presented go beyond
what we intend to accomplish with that system.

ARCHITECTURE
An important component of the OVIS2 system is a
discourse representation which is maintained as a
central data structure. At any time in the dialogue, this
data structure represents all that has been said so far.
It represents all utterances of both the system and the
user at several levels of abstraction. We distinguish a
syntactic level, a semantic level, a pragmatic level
and an information level. At the information level, the
dialogue is represented as an information exchange.
The information exchange consists of a number of in-
formation-state changes called updates. The informa-
tion state represents all the information that intention-
ally or unintentionally has been conveyed by the user
and also all information that the system has presented.
Almost all the modules within the system operate on
at least one level of the discourse representation. The
architecture that determines the relationships be-
tween these modules is depicted in figure 1. When the
user talks to the system, the incoming utterance is an-
alysed by a speech-recognition module, the result is
processed linguistically and then interpreted. While
this is done, the discourse representation is extended
with analysis results. Then the dialogue manager
comes into play. It determines what actions the sys-
tem undertakes at that particular point in the dialogue.
When these actions relate to things that the systems
should say, they are formulated and expressed in a

2. OVIS is a Dutch acronym for Public Trans-
port Information System. OVIS1 is based
on a system designed at Philips GmbH For-
schungslaboratorien Aachen (Aust et al.,
1994; Oorder and Aust, 1994).
linguistic form, and finally made audible.

In this paper, we focus on the pragmatic-interpretation and dialogue-management modules. The role of the pragmatic-interpretation module is to deal with implicit information, that is information that is not explicitly mentioned. Humans often convey information by not saying something. For instance, when users say: "I'd like to travel Tuesday", then, in most common contexts, one can deduce from the fact that they did not say which Tuesday they want to travel, that it will be the first Tuesday to come. Natural language dialogues are full of such implicit information transfers, which are possible because both participants share knowledge about the world. Therefore, we use a pragmatic interpretation module that enriches the results of the linguistic processing by incorporating knowledge about the world.

The dialogue manager will be presented with several, possibly conflicting, interpretations of what the user has said. It has to decide which of these interpretations should be pursued in the dialogue. The other interpretations are held back and may be used as backup for the case that, later in the dialogue, the first interpretation turns out not to be what the user intended to convey. The interpretations are represented as updates of the information state. The dialogue manager decides, on the basis of the updated information state, what the system should say next. This is expressed in another update, that is passed to the generation modules.

**UPDATES**

Information states contain objects and relations. These represent the information that is conveyed during the dialogue. In OVIS2, there are objects and relations for the origin and destination of a train connection, for date and time specifications, Amsterdam Central Station, five o'clock and tomorrow. In the dialogue, every utterance updates the information state. This means that objects and relations are added or removed. Utterances are analysed as instructions that determine how the information state should be updated. The update has an informational content, that is represented by objects and relations, and furthermore, we identify communicative functions and a ground-focus structure to represent the change character of the update. The ground in an update identifies the objects and relations that need to be changed, and the focus, together with the communicative function determines the change itself. The ground corresponds to things that the utterance is about, whereas focus correspond to what is said about them. In Valliduvi (1990), an overview of several notions associated to ground and focus is presented. Here, in this paper, we think of ground as those parts of the update that are already present in the information state, and the foci as those parts that describe a change of information.

As a catalogue of all the different kinds of objects and relations in the domain that we want the dialogue system to converse in, we define a frame structure. A frame structure is a set of types that are related by fea-
tures. The frame structure for the OVIS domain is shown in figure 2. The black dots represent types, and the lines with labels represent features. At the top of the frame structure is a type that is used for the object that represents the entire domain, or the "world". The fact that a world object is related to an object that represents the user, is represented by a user feature between the corresponding types. In turn, the type that contains users has subfeatures for representing where the user is, and what the user wants. The user may want to be somewhere else, he or she may want to travel, get information, browse through a number of possible connections, enquire about more connections or quit the dialogue. Each of the corresponding features leads to subordinated types. This goes on until, at the bottom of the frame structure, aspects are so small that they can be represented by a mere truth-value. Truth-values have a type that is shown at the bottom of the frame structure.

A frame structure is a rudimentary description of the domain. It describes the relevant objects and relations, but it does not describe the dependencies between relations. For instance, the fact that a date with a given year, month and day, the week and the day of that week are uniquely determined is not represented in the frame structure. Also, the fact that Amsterdam

Figure 2: The frame structure for OVIS2.
Central Station is situated in Amsterdam is not represented. Furthermore, there are a lot of boolean features that are mutually exclusive, for instance, an hour that is related to the true object through a feature with name 10 cannot be related to the true object by features such as 3, 7 or 12. At first glance, it may seem a little far fetched to represent objects that can be identified by means of constant names with a whole range of features (one for each possible constant of the corresponding type) that lead to boolean type. The advantage of this approach is that we don’t need to distinguish between features and constants, and this greatly simplifies the way in which updates can be specified. In updates there is the distinction between ground and focus features. Any feature in the frame structure can play either role, and therefore, we need a frame structure that is not prejudiced.

In its simplest form, a ground represents a pair of objects between which at least two features could exist according to the frame structure. A focus is then the feature that is to be inserted between the objects. In general, however, things are somewhat more complicated. For instance, the ground can represent ‘the moment at which the user wants to travel from Eindhoven to Amsterdam’. To define updates, we start with an example: The assertion that the destination town is Amsterdam is described by the following update.

\[
\text{destination.town.} \{ : \text{amsterdam} \}
\]

Here, destination.place.town.\{\} is the ground, amsterdam the focus, and the colon (‘:’), which stands for assertion, is the communicative function. The update is efectuated by finding an appropriate pair of objects, and inserting a feature \text{amsterdam} between them. The appropriate pair \(a, b\) will satisfy a number of criteria: Firstly, it must be of an appropriate type, so that \text{amsterdam} can be inserted. Secondly, \(a\) must be subordinated to some object \(a'\) by means a \text{town} feature. And the object \(a'\), in turn, should be subordinated to another object \(a''\) by means of a \text{destination} feature. In this case, the object \(b\), must be the truth object \text{true}. This is implicit—the update itself only specifies that it must be an object of boolean type—selecting \(b\) to be \text{true} rather than \text{False}, represents the fact that the utterance expresses positive information.

Another example shows that the frame structure should not be prejudiced as to which features are focus and which belong to ground.

\[
\text{user.wants.travel.} \{ : \text{destination}.amsterdam \}
\]

Here, destination is focus, while \text{amsterdam} is part of the ground. This update signifies that Amsterdam, which somehow was mentioned in the previous discourse is to be regarded as part of the destination description.

Due to the vagueness inherently present in natural language, the objects that mark the ground-focus boundaries will often not be uniquely defined. The pragmatic interpreter will use the dialogue context in order to select the most likely candidates.

The communicative function ‘\?’, in the examples above, corresponds to an assertion. There are also other communicative functions, for instance, ‘\!’ for corrections, ‘\*’ for confirmations, ‘\#’ for denials, ‘\-’ for answers and ‘\?’ for questions. Each of these communicative functions has a different influence on the information state. For instance, the correction operator ‘\!’ assumes that there is already some information, that is contradictory to the given focus. The new information overrides the old one.

Updates contain presuppositions. The first kind of presupposition is that the ground of the update is already present in the information state. So, we can express presuppositions as in the following example.

\[
\text{user.wants.travel.}
\begin{array}{l}
(\text{origin.eindhoven} ; \\
\text{destination.amsterdam} ; \\
\text{moment.} \{ : \text{clock_hour.10} ; \\
\text{minute.30} \} \\
) \end{array}
\]

Anything but the time specification (between square brackets is ground). The ground stands for ‘the moment that the user wants to travel from Eindhoven to Amsterdam’. The presupposition is that the user actually wants to travel from Eindhoven to Amsterdam. Another presupposition of this update is related to the communicative function. In this case an assertion. Assertions presuppose the information given in the focus was not known beforehand. Other communicative functions have different presuppositions. For instance, a corrections presupposes that there is some information that is incorrect, and an answer presupposes that something was asked for.

**PRAGMATIC INTERPRETATION**

Updates of the information state are used to represent the explicit meaning of user utterances. However, there is more to meaning than meets the eye—when the user says: “I am at Utrecht Central Station, and I want to go to Groningen”, then the fact that the user calls from a railway station can be exploited to set the time, date and place of departure. This is particularly important to avoid the system asking: “From where do you want to go to Groningen?” or “On what day do you want to travel?”. It is the role of the pragmatic interpreter to make this kinds of pragmatic inferences. For an overview of pragmatics see Levinson (1983).

The information that is explicitly mentioned by the user is often very sparse—in many cases, for instance in elliptical utterances, only a focus is given. Sometimes, linguistic means, such as parallelism, can be
used to enrich this information, but this is not always the case. Therefore, the pragmatic interpreter must be able to identify the objects between which information is to be updated using only very little information. For this purpose, a topic stack is maintained. The top of this stack represents the current topic. Updates are interpreted relative to the current topic. Both the user and the system influence the current topic by explicitly mentioning it (Grosz and Sidner, 1986). We will give an example: Suppose that in a context where the system just asked: "From where to where do you want to travel?", setting the current topic to the travelling of the user, and the user responds "From Amsterdam". Then this utterance is translated into the following update

\[\text{origin.place.}[= \text{amsterdam}]\]

Given the current topic, the ground can be identified uniquely, even though there may be several pairs of objects that qualify as ground-focus boundaries for the specification \text{origin.place.}[]. This particular pair is pushed onto the topic stack and the relation \text{amsterdam} is added in between. The pragmatic interpreter deduces some more things: Firstly, Amsterdam is a town, and therefore, the following update is added relative to the given place object.

\[\text{town.}[= \text{amsterdam}]\]

And secondly, it is very likely that the user will want to go to the central station in Amsterdam, otherwise he/she would have stated differently. The update \[\text{station.}[= \text{amsterdam}_c]\], and consequently, \text{id.amsterdam}_o and \text{suffix.o}, are generated to reflect this. Suppose that the system wants to verify this, then it can simply ask: "From Central Station?". This is because the objects that correspond to the paths \text{origin.place} and \text{amsterdam} are in the current topic, relative to which the phrase "Central Station" can be understood. Suppose, instead, that the system does not want to verify, but asks: "When?". Now, the place is no longer topic, so it is popped from the stack, and then the new topic is pushed. This topic is the moment that the user wants to travel from Amsterdam. In the update formalism, this is denoted as

\[\text{user.wants.travel.} \text{origin.}(\text{place.amsterdam ; }[? \text{moment.}])\]

When the user answers "At eleven o'clock". The corresponding update is

\[= \text{at.clock.hour.11}\]

The user's answer consists of focus only, and is interpreted relative to the current topic. Thus it is interpreted as

\[\text{user.wants.travel.} \text{origin.}(\text{place.amsterdam ; }\text{moment.}[= \text{at.clock.hour.11}]\]

The use of a topic stack facilitates an efficient dialogue, because lots of information does not need to be presented explicitly. To exploit the topic-dynamics, the dialogue manager should choose a suitable order in which it inquires about topics. This will be described in the next section.

**DIALOGUE MANAGEMENT**

The main concern of the dialogue manager is to determine what the system will say, and when to query the database. Furthermore, it should facilitate a natural and efficient dialogue in which the user can easily get the required information. In doing this, the dialogue manager has a number of problems: The systems assessment of the dialogue situation is littered with uncertainties. This is partly due to the technology, which is not fully developed, and partly an inherent quality of spoken dialogue. The current state-of-the-art in speech recognition and understanding does not allow for error-free processing of the user's utterances. And furthermore, pragmatic interpretation is inherently no better than educated guess-work. Even if the dialogue situation is assessed correctly, it is often hard to decide what is a good next move. This is because the decision relies on a prediction of the future dialogue, which is hard because the user's behaviour is largely unpredictable. Because of these uncertainties, the dialogue manager has to engage in a game of guessing and verifying—always trying to get more certainty about what was understood.

Even in human-human dialogues, there is never complete certainty that what one intends to convey is actually understood, or that what one understands is what the other participant intended to convey. Rather than regarding this to be a problem, one could state that this is one of the reasons that people engage in dialogues in the first place.

To cope with the uncertainty that the user may not understand the system, we can make the system generate quite elaborate utterances, so that we can assume they will be sufficiently clear to the user. This leaves us with a system whose only uncertainty is that it can never be completely sure that its analysis of the meaning of user utterances is accurate.

Nevertheless, we think that dialogue systems are feasible, because, as can be observed in human dialogues, complete certainty is not necessary. What we do need is a way to assess and manage the uncertainties. It is a joint responsibility of the system and the user to engage in a useful dialogue. The system's objective is to be a reliable and cooperative dialogue participant. What this requires is that relevant uncertainties are made explicit, so that the user may respond to them. The system may do this by asking
questions, and by explicitly mentioning what it "thinks" that it understood.

The dialogue manager focuses on the user's wishes and checks whether it can fulfil them. The user may want to get information about a train connection, or to browse through a set of connections that is already specified, or to quit the dialogue. When the dialogue manager can initiate some action to meet the wishes, it does so. But when it cannot, then there is also a reason for action, either to notify the user of the inability, or to ask the user for information that will help to meet the user's wish. The dialogue manager maintains an agenda of items that must be dealt with in order to meet the user's wishes. There are several kinds of items on the agenda. Firstly, there are items that represent database slots that must get a value before the system can give information about a connection. Secondly, there are items representing uncertainties that must be clarified. Then, there are items that correspond to inconsistencies, for instance, when the update date [1st month, February; day: 31] is processed. Then there are ambiguities, leading to disambiguation items, and finally, there may have occurred errors in the recognition, for instance, because the user speaks too softly. The dialogue manager selects an item from the agenda and this will determine what the system will say next. Selection of items from the agenda must be done with care. For a natural dialogue, clarification of uncertainties in the current topic must be dealt with before moving on to another topic. Exactly which scheme is best here, is still subject of research. In the following subsections, we will discuss the various kinds of items that may occur on the dialogue manager's agenda.

Errors
Because the recognition capabilities will be considerably more modest than can be expected from a human dialogue participant, modesty is required of the system. Therefore we assume that errors are always related to some incapability of the system. For instance, the speech recognizer may not be able to recognize words when they are spoken in a volume that is either too loud or too quiet. In that case, it will send an error message to the dialogue manager. This will cause an error item to occur on the agenda, causing the dialogue manager to act as follows. Firstly, it will apologize to the user, then it will notify the user what the problem is, and if possible it will give a suggestion to the user, to change his/her behaviour in a way that may circumvent the problem.

Uncertainties
To deal with uncertainties in the understanding of user utterances, the interpretation modules may generate several interpretations for one user utterance. These interpretations are associated with confidence measures that depend on acoustic and linguistic probabilities. The confidence measures are also reflected in the information state. So, the objects and relations are associated with confidence measures. Information with very low confidence measures will not have a lot of influence on the continuation of the dialogue, but when the measures are higher, but not too high, they may trigger the dialogue manager to verify the corresponding information. Verification can be done by mentioning the information explicitly in an utterance. For instance, when the user says: "To Amsterdam", the interpretation components may set generate the update destination.town.[: amsterdam]. When the confidence in this interpretation is low, the system may ask a question to verify it, as in: "Did you say you wanted to go to Amsterdam?". When the user answers this question positively, the confidence measure can be increased.

Inconsistencies
With cooperative users, and a simple domain such as train connections, inconsistencies are a rare phenomenon. Therefore, the assumption can be made that the user did not attempt to convey the inconsistency, but that it arose due to recognition errors. Under this assumption, interpretations that contain inconsistencies can simply be discarded. However, when there are no alternative interpretations of the user's utterance, this 'solution' will not work. Thus in some cases, but only when the confidence measures are high and the alternative interpretations have much lower confidence measures, the dialogue manager will report the inconsistency to the user.

Ambiguities
Confidence measures are also important in order to deal with ambiguities. The dialogue manager can deal with them in several ways: It can ask an explicit question in which both alternatives are mentioned explicitly. This is, however, only helpful if ambiguity is not due to acoustic problems, because in that case, it is likely that the same problem will be present in next user utterance also. For instance, it is notoriously difficult to distinguish the two city names 'Baarn' and 'Maarn'. So if the user wants to go to Baarn, we may get an update such as the following:

destination.town.[= (.61) baarn | (.63) maarn ]

This means that the destination town is either Baarn, with confidence measure 0.61, or Maarn with a confidence measure 0.63. If the system would now asks the question: "Did you say you wanted to go to Baarn or to Maarn?" and the user responds with "To Baarn", 
then the same problem will arise again—an update such as the following will be likely.

```
destination.town.[ [ ( .59 ) Baarn | ( .62 ) Maarn ] ]
```

However, if the system asks instead: "Did you say you wanted to go to Maarn?", then the user will respond with something like "No, to Baarn". And although acoustically, Baarn still cannot be distinguished from Maarn, the "No" signifies that the user is denying that which has been mentioned explicitly in the system utterance. So apart from the update above, also an update that denies Maarn as destination town will be generated.

```
destination.town.[ # Maarn ]
```

Because of the denial, the confidence measure for Maarn will be lower than that of Baarn.

In cases of ambiguity that are not due to speech-recognition problems, an explicit question may be useful. For instance, if the user says: "I want to travel tomorrow, at 10 o'clock", then it may be unclear whether the time reference is am or pm. This may be asked explicitly, by means of "In the morning or in the evening?". But still, selecting one alternative and verifying it seems to be more natural (as in: "In the morning")

**Database Slots**

When the user wants to query the database, then a number of slots must be filled in a database query. The slots give rise to an initial question, that could, for the OVIS system, be formulated by the following question update, were it not that this would lead to a question that is far too complex for a single utterance.

```
[? user.wants.info.
  ( connection,
    ( origin.place.station.id._ ; destination.place.station.id._ ;
      _ .moment.( at.date._ ; after.time._ ; before.time._ ) ) ;
    quantor._ )
]
```

Therefore, the question is divided into several sub-questions, that are added to the dialogue manager's agenda as database-slot items. For OVIS, these slots represent an origin and a destination station, a date, a time interval, a reference that says whether the time interval refers to the departure or the arrival time, and lastly a slot that determines whether all connections should be returned or only the first or last one in the interval.

The way to get values for these slots is to ask questions about them to the user. By asking a question, several things happen. Firstly, the user is notified that the system does not know the slot's value, and that it needs a value to perform the required action. Furthermore, the slot becomes topic of conversation, so the user can respond in a natural way, with the required information as focus.

**Combining Items into Topics**

When information with sufficient confidence is gathered about the current topic, the dialogue manager will look on the agenda for another topic, which may well be a database slot. However, even when there are still some uncertainties, it is sometimes fruitful to change the topic in a way that still verifies the uncertainties. This can be done by using presuppositions. To give an example, consider the case where the user has just mentioned the cities Nijmegen and Groningen as the origin and destination, and where the confidence measures are too low to just accept the corresponding values, but high enough to take them seriously. Now, the dialogue manager may want to verify the origin and destination, but does not want to explicitly ask for verification. In such cases, the dialogue manager may introduce a new topic as follows: "When do you want to travel from Groningen to Nijmegen?". This corresponds to the following update.

```
user.wants.travel.
  ( origin.place.groningen ; destination.place.nijmegen ;
    [ ? moment._ ] )
```

Here, the origin and destination are explicitly mentioned in presuppositions. So that the user will know what the system thinks are the origin and destination. If this is incorrect, the user can still take action to correct the misunderstanding. But if the assumption was correct, the topic is effectively changed to the moment slot. We should, however, take care not to use this kind of verification to easily, because users may get confused when they are confronted with false presuppositions. But in cases where the confidence measures are relatively high, they may lead to a more efficient dialogue.

**SUMMARY**

We have described the design of pragmatic interpretation and dialogue management modules for spoken language systems. The modules are based on an update logic that reflect the topic-focus structure of the discourse. As the logic supports the vital issues at hand, such as uncertainty, ambiguity, underspecification and contextual interpretation, without too much over-generation, we regard our approach to be a promising one in dialogue system technology. We do, however, realize that our approach has limits. In more complex domains, in which a more flexible dialogue is needed, it may not be possible to express the mean-
ing of user utterances in terms of a rigid pre-defined frame structure.

ACKNOWLEDGEMENTS
This work is financed by the Netherlands Organization for Scientific Research (NWO).

REFERENCES
Grosz, B.J. and Sidner, C.L. (1986). Attention, Inten-
tions, and the Structure of Discourse. Computational Linguistics, Vol. 12, Nr.3, July-Sep-
tember 1986, 175-204
Levinson, S. (1983). Pragmatics. Cambridge Univer-
type of an Automatic Inquiry System. Pro-
Topics in SCHISMA Dialogues

Joris Hulstijn, René Steetskamp, Hugo ter Doest, Stan van de Burgt and Anton Nijholt
{joris|steets|terdoest|stan|anijholt}@cs.utwente.nl

ABSTRACT

An important part of dialogue management in a dialogue system is topic management: The system has to keep track of topic aspects (current topic, context, obligations, ...) in order to be able to resolve ambiguities and to respond in a cooperative and sensible way.

In the SCHISMA project, we are developing a system that takes part in a conversation about theatre performances and is which able to make reservations for these performances. In this paper we discuss several techniques for topic management that we developed and tested this year.

1 INTRODUCTION

In a (spoken) dialogue system one of the main tasks is answering questions that the user states, either explicitly or implicitly. As the question is often based on, or referring to, earlier utterances of either interlocutor, analysing the last user utterance will not suffice in most cases. This conjecture is verified in the SCHISMA project, a cooperative project of Twente University and KPN Research in the Netherlands. In this project the SCHISMA system is developpe: this system provides the user with information on theatre performances, and optionally sells the user one or more tickets for a given performance (Hoeven et al. 1995). The system started as a pure, human driven, Wizard of Oz (WoZ) system to which several additions were made as time went by and more subsystems became available. This way, the system evolves to the final system we envision. Currently, parts of the generator and the dialogue management modules are incorporated, while pre-processing modules will be incorporated later this year.

In an earlier phase of the project a corpus of dialogues was collected, which serves as a basis for much of the work presented in this paper. A dialogue is considered here to be a sequence of turns of two speakers: the user and the system. A turn is an uninterrupted sequence of words by one speaker. A turn consists of one or more utterances: linguistically identifiable units. Typically, the typed utterances in our corpus are marked by punctuation and conjunctives.

As we noted above, answering a question involves more information than can be found in the last utterance alone. In fact, we argue that the response is mainly determined by the following parameters: \(\text{information context, topic, utterance type (UT), expected response type (ERT) and dialogue phase.}\)

The \(\text{information context}\) incorporates both information extracted from the dialogue and domain knowledge. We choose to organise the context around the notion of topic, the entity the dialogue is currently about (see section 4).

Other parameters that guide the behaviour of the dialogue manager are \(\text{utterance type (UT)}\) and \(\text{expected response type (ERT)}\). The utterance type is roughly the grammatical type of sentence. (e.g. declarative, nominal, ...). The expected response type indicates what the user is expects the system response to be about. For instance, a \(\text{when}\) question from the user will prompt the system to respond with a notion of time or date.

Like topic, these parameters can be detected using syntactic cues from the user utterance, together with domain knowledge and the previous topic. This is discussed more thoroughly in section 6.

We distinguish several phases of the dialogue. Apart from the obvious \(\text{opening- and closing}\) phases, we distinguish a \(\text{information, a selection, a reservation and a confirmation phase.}\) During the first two phases, the information and selection phases, the user has the initiative.

For example, first the user requests information about performances, which is subsequently provided for by the system. During the reservation phase, the system takes more initiative, ask-
ing the user for information like name and address in order to be able to make a reservation. After this, the system lists the reservation details (performance, price) and asks for confirmation.

The reservation phase presupposes that a particular performance is selected. Note that there is no obligatory sequential order, just a logical one. So, it is very well possible for a user to go through the first three phases in one go, for instance by asking I want to reserve a ticket for tonight. Now if the information provided by that utterance is not enough to determine a unique performance, the system takes over initiative, asking the user for more details. So, the phase of the dialogue has a profound impact on the response behaviour of the system.

The architecture of the SCHISMA dialogue manager has progressed from a finite state-based to an utterance-based architecture. One of the reasons for this development is that the huge number of states for a nontrivial dialogue automaton, makes manual construction unfeasible (Bos 1995); some general principles are needed to automatically generate the automaton. Once principles are used, there is no longer a conceptual need for finite-state techniques\(^1\). This paper describes the first step in determining useful principles for our domain. Some of them will be applicable to any information-dialogue. But some will be particular to our theatre domain.

In section 2 we introduce the architecture of our system, followed by the principles of the grammar in section 3 and the role of context and topic in section 4. The implementation is explained in section 5. In section 6 we report on some of our ongoing corpus-based research into ways of determining utterance type, topic and expected response type. Section 7 raps up with conclusions.

2 ARCHITECTURE

The SCHISMA system basically consists of three modules: a morphological analyzer, a parser and a dialogue manager (figure 1).

The user produces an utterance, which is scanned by the morphological analyzer, MAF, and transformed by the parser into a meaning representation in the form of an item-list. An item list is a list of information items\(^2\), combined with their values and flags indicating the status of the information (see section 5). The parser uses a unification based grammar. We believe that the design of the grammar should focus on performance, not linguistic competence. Therefore, the grammar should ideally produce at most one parse. In order to achieve this the grammar is based on four principles, which are explained in section 3.

The dialogue manager determines the appropriate response action based on the meaning representation of the user-utterance and on the context. The dialogue manager also updates the context with the information from the user utterance. A response action may be a combination of three kinds of actions: a search in the database followed by a generated answer, a request for more user-information, or a control utterance like ‘thanks’ or ‘good bye’.

Item-lists are used to represent the information that the parser has been able to extract from the user-utterance that is relevant for the dialogue manager. Information that is not recognized by the lexicon or that cannot be parsed is neglected. Think of an item-list as a simplified version of the feature structure produced by the parser.

3 GRAMMAR

In developing a grammar and parser for the SCHISMA system, performance (contrary to competence) is the central issue. In our approach linguistic knowledge and theory is applied where necessary: we consider linguistics as a means for (domain-specific) meaning extraction rather than
a goal in itself. In literature this type of language modeling is sometimes referred to as semantic grammar.

Our parsing system should process any input, i.e. tackle the no-answer problem, and extract as much domain-relevant information as can be recovered without the dialogue context. So, the parser should map its input onto preferably one expression, representing the context-independent part of the meaning that the user wants to communicate. See for instance the feature structure produced by an adjunct like example (1). The schisma part will be turned into an item-list. The context will reveal the function of the adjunct, for instance as the continuation of a previous question.

(1) en op donderdag ? (011) and on thursday ?

Z:
[ cat: Z
  head: [ first: [cambesubj:-
  prep: OP ] ]
  schisma: [ mode: [questionmark:+]
  time: [week: [thursday:+]
  coordination: [OP]] ] ]

The one-parse requirement may seem rather trivial. However it has some implications that are less trivial and have clear consequences for the system as a whole. First, we need a dialogue manager that interprets the context-independent expression in context of the dialogue (see section 4). Second, if no alternative parses are allowed, introduction of lexical and structural ambiguity is allowed only if it can be accounted for at a later stage in the parsing process. Third, the parser should indicate exactly where contextual parameters still need to be filled in (for instance for anaphoric expressions or ellipsis).

We have translated the above findings into the following principles for the SCHISMA grammar development process.

P1 Do not introduce more (syntactic and semantic) alternatives in the lexicon than can be accounted for during parsing.

P2 The grammar should be as structural unambiguous as possible. Structural ambiguity is allowed only if, at a later stage during parsing, out of the introduced alternatives a selection can be made (using syntactic and/or semantic constraints). Note the correspondence to P1.

P3 Syntactic constraints guide the building of the meaning representation (semantic disambiguation).

P4 Semantic constraints guide syntactic processing (syntactic disambiguation).

In the course of writing the SCHISMA grammar the following two questions kept coming up.

Q1 What structure of utterances is important for determining the meaning of the input (and thus, what structure is unimportant)?

Q2 How can semantic knowledge about the domain be used to guide syntactic processing?

With respect to Q1, the best way to find out is to experiment with different set-ups. For these experiments in writing a SCHISMA grammar, we use a left-corner parser for PATR-II, developed at SIL (McConnel 1995, pcpattern). It is a straightforward implementation of PATR-II (Shieber 1986), and it can be easily incorporated in the SCHISMA system.

As for Q2, we are planning corpus-based research into subcategorization patterns and corresponding semantic roles in our domain. We realize that the grammar not only combines words, but also the concepts behind it (Sowa 1984). Domain specific information can be applied in a typed unification-based grammar. Both linguistic and conceptual knowledge are represented in type-hierarchies. In (Steetskamp 1996) such a semantic parser for a fragment of the SCHISMA domain is described. It makes use of the head-corner parser generator for typed feature structures that was developed at our department (Moll 1995).

4 CONTEXT AND TOPIC

Our corpus of information dialogues contains heavily context-dependent types of utterance. Answers depend on questions, anaphora depend
on their antecedent and elliptical expressions depend on the previously uttered phrases. See examples (2), (3) and (4) respectively.

(2) S: Hoeveel kaartjes wilt u? 
   How many tickets would you like.
   U: 4

(3) U: Daar wil ik wel heen.
   wanneer spelen zij?
   I'd like to go there
   When are they performing?
   S: Op 7 januari kunt U naar
   "Cocktail".
   On the 7th of januari you can see
   "Cocktail"

(4) U: En Othello?
   and Othello?

We believe that the information structure of the context is just as important as its information content in resolving such dependencies (Chafe 1976). The information structure of the dialogue context is modelled using topic. The topic specifies what the dialogue at that point is about. With each topic some information items are naturally associated. For instance, the topic performance often co-occurs with a date, an artist or group and a performance title (see figure 2). So, the topic specifies a frame of information-items associated with the topic.

Some items function as subtopic. Subtopics have a frame of their own. So, we get a hierarchical frame structure. The current frame structure is isomorphic to the design of our database.

So, the frame-structure that specifies the relations between potential topics, subtopics and attributes is our way of incorporating domain-knowledge in the system. However, a frame-structure cannot express all that needs to be known. For instance, the fact that a reservation presupposes a performance to have been selected, cannot be represented in the frame.

There are three other points in the architecture where domain-knowledge plays a role: in the grammar during unification (see principle P4), in the procedure that determines whether an itemlist matches a frame (section 5) and in the procedure that specifies the actions needed to make a reservation. At the moment we have no formal way of specifying this kind of procedural knowledge. One way, would be to have both user- and system plans (Allen et al. 1991), to guide the system's behaviour and help understand the user's behaviour. Some of the intuitions behind plans are coded in our dialogue phases.

4.1 Other Approaches to Topic

The notion of topic is problematic in the literature. Different traditions use it in different ways. Our notion of topic compares best to the notion used by Mieke Rats in her analysis of a corpus of information dialogues7 (Rats 1996).

An entity T is the topic of an utterance U if U is intended to increase the addressee's knowledge about T, request information about T or otherwise get the addressee to act with respect to T ((Rats 1996, p 37)).

Our use of topic differs from other well-known approaches. First, it is not a question-based approach (Kuppevelt 1995). Kuppevelt uses a notion of topic based on aboutness. According to Kuppevelt, topic is intimately related with the most salient question that needs answering at that point in the discourse: what question is it that makes the sentence relevant at this point? The question may have been asked explicitly, as in our information dialogues, or it may be there implicitly. Topic is defined as the (type of) entity that would provide an answer to that question.

---

6Topic-focus related ideas have been used widely in dialogue systems. Recently, (Veldhuijzen van Zanten 1996; Smith et al. 1995; Deemer et al. 1994; Rats 1995). See section 4.1 for comparisons.

7The definition is based on (Gundel 1986).
We found that the question-related notion of topic, although theoretically appealing, did not provide enough detail for successful automatic application to the corpus. Moreover, wh-questions comprise over a quarter of the corpus, so they are important enough to introduce a separate notion of topic to account for them. This is why we have *expected response type* among the parameters that determine the dialogue manager's behaviour (section 6).

Second, our notion differs from the Prague school approach (e.g. (Hajicova et al. 1995)). The Prague school assumes a strong link between topic and contextually given information. New or contrastive information, on the other hand, is said to be in focus. Elements in focus often get the sentence accent and are generally placed at the tail-end of the sentence. Changing the word order (for instance by left dislocation or topicalization) changes the information structure. Example (5) (constructed) illustrates the differences.

(5) S: U kunt naar de cabaret voorstellingen door Herman Finkers en 'Commil Foo'.
     You can see the comedy shows by Herman Finkers and Commil Foo.
U: Herman Finkers, wanneer is die in de schouwburg?
     Herman Finkers, when he in the theatre?

The left dislocation in the user utterance, indicates that the fronted *Herman Finkers* is in focus. So, that fits nicely: picking Herman Finkers marks a contrast with other names from the selection. On the other hand the question clearly is about Herman Finkers. So, the current topic is *Herman Finkers* too. So, topic (aboutness) and focus (informativity) represent separate dimensions. Now, the when question sets the expected response type to be a date. According to Van Kuppevelt, a date would be the expected next topic. So, suppose the system answers with a date, say 15th of May. Would we say that the answer was about Herman Finkers or about the date? To solve this, we concluded that expected response and aboutness express different dimensions too. The topic of the answer remains em Herman Finkers.

A third approach to these issues is the centering approach. Our notion of topic roughly corresponds to the *backward looking center*. Center-

---

\[\text{item-list} \] \hspace{1cm} \text{An item list is a list of item-value pairs headed by a tag indicating the utterance type (UT) and a list of items indicating the expected response type (ERT).}

\[\text{item-list} ::= [\text{UT} : \text{ERT}; \text{IV}_1, \ldots, \text{IV}_n]\]

where

\[\text{UT} ::= \text{DEC} | \ldots | \text{NIS}\]

\[\text{ERT} ::= _- | \text{Item}_i(\text{Items})^*\]

\[\text{IV}_i ::= \text{Item}_i = \text{Value}\]

\[\text{Item}_i | \text{Value} \quad (1 \leq i \leq n)\]

Item-value pairs may indicate what value an item has (\(-\)), what value an item will not have (\(!=\)) or that the item has been given no value (\(_-\)). In that case, the value should be provided by the context. There are 'unary' items indicating confirmation, denial, greetings and thanks. We assume that each item occurs at most once.

---

\[^8\text{See} (\text{Gross et al. 1995}) \text{for an introduction.}\]

\[^9\text{The center is sometimes called} \text{ focus of attention.}\]

\[^10\text{Dutch is like German in this respect.} \text{ (Strube and Hahn 1996)} \text{ have formulated a preference order for German, based on the semantic functions of constituents.}\]

\[^11\text{For a list of utterance types, see table 1.}\]
Item names are taken to be equal to the names of the slots in the database. Values are strings or numbers\textsuperscript{12}. Here are some examples.

(6) Wanneer zijn er musicals? (820)
   \textit{When are musicals on?}
   [ WHERE; time; genre = musical ]

(7) Ik wil niet naar de 'groene vogel' (2369)
   \textit{I don't want to go to the 'groene vogel'}
   [ DEC; ---; title != 'groene vogel']

In general there are two kinds of items: control items, like utterance type or thanks and domain items like title or date. Control items merely convey the form of utterance, domain items convey its content. There are no items indicating the expected topic or the phase of the dialogue. These are stored and determined within the dialogue manager. A topic shift often occurs when an item-value pair is not compatible with the current topic. A phase shift is often indicated by an empty ERT (section 6). In the future, the set of control items may be extended with other cues, for instance the subject type and verb type or the presence of a question mark.

Note, that the input to the dialogue manager actually is a feature structure produced by the parser. To get an item-list, the feature structure needs to be converted first. Although this conversion usually is straightforward, some information may get lost. One of the main reasons for this loss is the simplifying assumption that all relevant information can be converted into item-value pairs.

\textbf{context datastructure}

We need a representation of the context that respects the topic structure. Some of the characteristics of our topic structure are that it loosely resembles a tree, that the information along each branch must be consistent and that it must be possible to come back to previously discarded topics. The topic structure described here corresponds to a fragment of the frame-structure introduced in section 4. It is, as it were, a filled-in frame-structure.

The context datastructure is a list of which the elements have two components: a view (the active records in the database) and a set of item-value pairs. The information common to the records in view corresponds to the information that has been conveyed in the dialogue. Adding information means narrowing the view. When somebody selects genre=dance, the view is restricted to all performance-records that have dance in their genre fields (figure 3). The view is stored in a priority-list of active records from the database. All records in view are moved to the head of the list. Records that are no longer in view remain in the list outside the view. In this way, the view can be stored as an integer value: the number of records in view counted from the head of the list.

Note that the view mechanism only allows a monotonic increase of information. Backtracking is possible, but all information that has been added in between is lost from the view. This is undesirable, since some of the lost information may not be logically related to the discarded topic. Therefore, after backtracking, the view needs to be computed from scrap, using the list of item-value pairs (see also example 8).

Item-value pairs are stored with a flag that indicates its status. An item is wanted when it occurred in the ERT or when the system needs it in order to make a reservation (e.g. user address). An item is given when the value is present in the item-list. An item is mentioned when the item has come up in conversation but the value has not been supplied in the utterance.

A list of item-value pairs is said to match the current context, if (1) all mentioned items occur in the frame of the current topic, and (2) for all given items, there is at least one record in the view with corresponding values. When the view turns out to be empty, we say the item-list does

\textsuperscript{12}For times and dates we use the representation formalism developed by (Aust and Oerder 1984).
not match\textsuperscript{13}.

Suppose that after updating the view with restriction \texttt{artist='Jules Deelder'}, the view has become empty, that means that 'Jules Deelder' did not match in the last context. In that case we would need to backtrack to a previous node on the context list, where 'Jules Deelder' would in fact match.

backtracking

Here is an example of a dialogue with complicated topic-structure \textsuperscript{14}.

(8) U: What's on at the 18th of march? 1
    S: On the 18th, you can see 'Deelder Denkt' or 'Indonesian Tales'.
    U: At what time does Deelder start? 3
    S: The performance by Jules Deelder starts at 8 o'clock.
    U: How much is it? 5
    S: We give student discount. Do you have a student card?
    U: No 7
    S: The ticket for 'Deelder Denkt' without discount is 390,-.
    U: Oh, that's expensive. 9
    U: How much are the Indonesians? 10
    S: One ticket for 'Indonesian Tales' without discount is 390,-.
    S: Give me Deelder then after all. 12

After mentioning the date, 18th of march, there are two alternatives open. The user can freely switch between them. In sentences 3 and 4, the user asks information about Jules Deelder. This limits the topic to the Deelder performance. Discount is a subtopic of that. However, in utterance 10 the user switches to the other alternative. This will happen, because the phrase 'Indonesians' does not match with the discount, nor with the Deelder context. It does fit the '18th of march' context, producing a new frame.

The discount information seems to be lost. However, discount not only is a subtopic of a performance (via price) but also of the control topic user. Since this kind of user information is available in all contexts, the information is not lost. However, the time information does get lost in backtracking. This is as it should be because the 8 o'clock time is associated with the Deelder

\textsuperscript{13}The matching procedure would be the place to extend the system with a domain inference module.

\textsuperscript{14}The example is constructed, but based on utterances (903-958).

![Figure 4: Topic Structure of (8)](image)

performance. So apparently there are two distinct realms: facts about performances and user-information.

generation

We use sentence templates for the generation of system utterances. It is well-known that automatic sentence generation produces a more natural flow of dialogue when topic-focus issues are taken into account (e.g. (Deemter et al. 1994) for spoken monologues). The templates are stored in a table that can be indexed by an expression that is combination of template type (doc, whq, yq), control items, (yes, thanks, bye), items that are to be presented as given and items that are to be presented as new. Templates are constructed in such a way, that given items appear up front. New items may appear topocalized, or at the end of the sentence. The new items will be the results of the database search for items in ERT. The given items will normally refer to the topic.

We plan to experiment with different formats. For instance, we could choose to always use demonstratives or pronouns to refer to given items. We could choose, to always respond to a wh-question with its ERT only. The hypothesis
is, that this would reduce the complexity of the dialogues.

6 Determining Topic, UT and ERT

This section deals with the following question: how can we automatically determine the topic, the utterance type and the expected response type (ERT), given the information in the feature structure produced by the parser, the previous utterance and the previous topic?

Utterance-type

Utterance-type is the grammatical type of utterance. It can be detected by the parser using surface information. Utterance type is related to the communicative function, but not equal. For instance, in example (9) below a declarative sentence is used to request information about a performance. In fact, the majority of the declarative user utterances in our corpus are of this type.

The set of utterance types in Table 1 is developed by (Anderman 1996). Anderman uses so-called cue-patterns representing surface information for automatic classification of utterances. The utterance type is an important cue, as are the presence of a question mark and the semantic type of subject and verb. The corpus has been manually annotated with these cues. Utterance type can be detected using principles of Dutch word order. In a normal declarative sentence the 1st position is the subject position. Wh-questions have a declarative word-order, but have a wh-word in 1st position(SVO). Yes/no questions have SOV as do topicalization constructions.

<table>
<thead>
<tr>
<th>Utterance Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHQ</td>
<td>fin verb on 2nd, wh-word on 1st</td>
</tr>
<tr>
<td>DEC</td>
<td>fin verb on 2nd, no wh-word</td>
</tr>
<tr>
<td>YNQ</td>
<td>fin verb on 1st, subject on 2nd</td>
</tr>
<tr>
<td>IMP</td>
<td>fin verb on 1st</td>
</tr>
<tr>
<td>PRE</td>
<td>prepositional phrases</td>
</tr>
<tr>
<td>NOM</td>
<td>nouns, nounphrases, proper names</td>
</tr>
<tr>
<td>ADJ</td>
<td>adjectives, adverbs or numbers</td>
</tr>
<tr>
<td>THA</td>
<td>thanks</td>
</tr>
<tr>
<td>GRE</td>
<td>greetings</td>
</tr>
<tr>
<td>CON</td>
<td>confirmation/negation (yes, no)</td>
</tr>
<tr>
<td>XCL</td>
<td>interjection, emotives, exclamations</td>
</tr>
<tr>
<td>MIS</td>
<td>miscellaneous</td>
</tr>
</tbody>
</table>

Table 1: Utterance types

determines the ERT. Similarly, a yes/no question may expect, apart from a straight yes or no, additional information. (10) Therefore all utterances can specify an ERT.

(9) Ik wil graag naar de voorstelling
    van Youp van 't Hek
    I'd like to go to the show by Youp
    van 't hek
    ERT: date

(10) Speelt Toneelgroep Amsterdam
    ook bij jullie?
    Does Toneelgroep Amsterdam play
    at your place?
    ERT: yes + date

An empty ERT indicates that the syntactic form of the sentence does not normally demand a response of a specific type. This is the normal situation during the reservation phase, when the system has the initiative. In the other case an empty ERT often means a phase-shift. (see below)

topic

There is a number of sources in the syntactic form of a user-utterance that indicate the current topic or introduce a new topic. We have no proper algorithm for detecting the topic. We do have a number of observations, that make it plausible that such a detection algorithm is feasible. First, there are general observations regarding the way topics are referred to. With respect to topic structure, user utterances can be divided in four groups, ordered by the relative complexity of the expression used to refer to the topic. (Compare the results of (Rats 1996, ch5)).
1. topic introduction: a new topic is introduced using definite descriptions, proper names, (fragments of) titles or complex referring phrases. Performances may be selected using date.

2. topic shift: a different topic is introduced. This happens when items from the utterance do not match the frame associated with the current topic.

3. topic narrowing: a subtopic of the previous topic is introduced. Uses the same mechanism as (2).

4. topic continuation: the topic remains the same. The topic is referred to using shortened descriptions, demonstrative pronouns, personal pronouns, ie (it, he) or the topic is simply left out.

Secondly, we have collected a number of detailed templates. The templates look like grammar rules. Each template represents a class of utterances of the same form. For each template the expected new topic and the expected response type are indicated. By way of example, the templates for declarative utterances and for wh-questions are presented in detail. Similar templates exist for the other utterance types.

declaratives

Our corpus contains 2414 utterances, of which 978 are user-utterances. Of a total of 130 declarative utterances, 79 contained the word wil (want) and 111 contained the word ik (I). Apparently most declarative sentences are of the form ik wil X (I want X), where X then refers to the new topic. The interjections graag (like to), toch (after all) and nog (still) occurred a lot. The polite form ik zou (graag) PS willen (I would like to (have) X) also occurred. (8 times) There are basically two classes depending on the type of X: utterances requesting information about a performance and utterances requesting tickets or a reservation for a performance. A small class consists of answers to a system-question. (see figure 5)

However, it is difficult to judge if the user wants information or already wants to reserve tickets. This can often be decided on the basis of the type of X. Is it a performance title, an artist, group or date, we assume that the user wants information. Is it related to tickets, reservation or price, we assume the user wants a reservation.

A particular performance or set of alternative performances can be introduced as topic, using genre, artist/group, title or date. When the user selected a (set of) performances by one or two of these items, the system ought to respond with the other items from the triple (group/artist | title | date). When all of these are given, the system still has the possibility of listing the text of a review or newspaper clippings.

The following notation is used: () indicates optional elements. | and / indicate alternatives. Words in capitals represent syntactic categories of the grammar.

wh-questions

There are 278 Wh-questions, which is more than a quarter of all client utterances. Some question words directly determine the expected response type. Wie expects a person. But others, like welke and wat need to be combined with other phrases. Welke NP expects an answer about or of type NP (example (11)). Wat is more difficult. Usually it combines with the expected object of the verb. For instance, in example (12) the expected answer is a price, because price normally is the object of cost.

(11) Welke opera’s worden gespeeld? (1349)
Which operas are performed?
ERT: opera

(12) Wat kost een kaartje voor die opera? (265)
What does a ticket cost for that opera?
topic: die opera = ‘topic
ERT: price

The exact syntactic category does not matter for determination of the topic and ERT. If the wh-phrase is the subject, the topic is often given by the object of the verb. When the wh-phrase is direct object or adjunct, the topic is given by the subject of the verb. In both cases it is the first PS that comes after the WHP.

**WH1** WHP V PS1 (PS2) ?
topic: PS1 \( \text{\&} \) ‘topic
ERT: type(WHP)

**WH2** wat V PS1 (PS2) ?
topic: PS1 \( \text{\&} \) ‘topic
ERT: type(1st subcat(V) \( \neq \) PS1).

---

17Opening and closing phrases by the system are all tagged separately. This explains the odd proportion.
D1 Ik wil (graag) naar genre | group/artist | title | date
   I want /like to go to ...
   topic = performance
   ERT  = if not given group/artist | title | date
         else __

D2 Ik wil / zou (graag) / information over genre | group/artist | title | date (wollen) (hebben)
   I want to / like to have information on ...
   topic = performance
   ERT  = if not given group/artist | title | date
         else review

D3 Ik wil / zou (graag) (N) tickets (voor group/artist | title | date) (wollen)(reserveren)
   I want to /like to have/reserve (N) tickets (for ...
   topic = tickets < performance
   ERT  = __

D4 Ik heb PS1
   I have PS1
   topic = PS1 | 'topic
   ERT  = __

Figure 5: Declarative Sentence Formats

7 SUMMARY

This paper explains some of the choices that have been made in the design of the SCHISMA dialogue manager. We cannot yet conclude that these choices are a success; the work is still very much in progress.

The design can be characterized as an utterance-based dialogue manager: the syntactic structure and content of the user utterance, together with the structure and content of the context, determine an appropriate response action. The parameters that determine the state of the dialogue and therefore the behaviour of the dialogue manager are phase, utterance-type, expected response type and a detailed datastructure that keeps track of the context.

The design of the grammar is based on principles, that help to have the parser produce one parse only. The idea is to keep the number of alternative parses down by limiting alternative lexical entries, by disallowing mere structural ambiguity and by combining syntactic and semantic constraints from the domain. Domain information can be applied in the grammar using type-hierarchies and a typed unification grammar.

The context datastructure is organized around the notion of topic: what the dialogue is about at that point. With a topic several information-items are naturally associated. This results in a frame-structure, that functions as the design of the database. Items are slots in a frame. values are filled in as the dialogue progresses. Consistency is checked using a database-view.

Organizing the context around topics in this way, has a number of advantages: context dependent utterances like utterances containing anaphora, answers to questions or ellipsis, can more easily be understood. Backtracking and correcting previous choices becomes possible, without discarding all information that has been uttered in the meantime. It remains possible to come back to previously discarded topics.

Our use of the notion of topic differs from a question-based approach (Kupsevelt 1995). Questions do have influence on the topic, but are better dealt with using a separate notion: expected response type. We differ from the Prague School (Hajicova et al. 1995) in that word-order is not used as the main cue to discern the topic. Rather we formulated a number of standard sentence formats that utterances in our corpus can follow. For each of these formats, the predicted effect on topic and response type was indicated. The results come out remarkably similar to the preference order used in centering theory (Grooss et al. 1995). Word order does make a difference in the selection of sentence templates that are used in sentence-generation. Templates are indexed using combinations of given and new items.
REFERENCES


DIALOGUES IN AIR TRAFFIC CONTROL

Gavin E Churcher, Clive Souter & Eric S Atwell
School of Computer Studies,
Leeds University, Leeds LS2 9JT
UK
Tel: +44 113 2336827
(Email: gavin@scs.leeds.ac.uk)

ABSTRACT

We have taken an off-the-shelf, commercial continuous speech recogniser and conducted evaluations for the domain of Air Traffic Control. The language of this domain proved to be quite unrestricted, contrary to our initial intuitions. Our experiments show that constraints typically used by speech recognisers do not provide accurate enough results and need to be augmented with other knowledge sources and higher levels of linguistics in order to prove useful.

We used three syntaxes based on a corpus of transmissions between the ATC and pilots in order to reflect differing levels of "linguistic" knowledge. Initial experiments demonstrate the benefit of a fully constrained context-free semantic grammar. Further experiments empirically show the benefit to recognition accuracy of using some form of dialogue management system to control the flow of discourse. A corpus-based statistical clustering approach to the segmentation of a dialogue into discourse segments is briefly discussed.

1. Instructions to the pilot to change his/her altitude. Information would be an altitude either in terms of a height in feet or a flight level.

2. Pressure settings for QFE (observed pressure) and QNH (altimeter/sub-scale setting). Pressure settings are measured in millibars.

3. Secondary Surveillance Radar (SSR) settings for squawk values. Squawk values are transponder settings which enable ATC to identify aircraft via radar.

4. Instructions to the pilot to change to another radio frequency.

5. Instructions to the pilot to change his/her heading, a setting measured in magnetic degrees.

Appendix I contains some example transmissions by the ATC; important information is highlighted.

The domain was initially thought to be complex, but practical, requiring continuous, speaker independent speech recognition with real-time response. In order to start building a model of ATC utterances, the Radiotelephony Manual [RTF CAP413] was examined. The manual provided protocols and examples for a number of situations such as landing, taking off, changing frequency etc. To have a better idea of the actual language used behind the protocols, a corpus of transmissions was collected.

It was this corpus (see The LBA Corpus below) which led us to believe that the ATC domain used choice phrases for each of the above areas which
could deviate slightly in many different ways. For example, instructing the pilot to change his radio frequency can start with phrases such as: "contact the tower now", "proceed to contact the tower on...", "you are free to call the tower..." etc. These key phrases were also interspersed and surrounded by other 'noise-phrases' representing other information and apparently free English language.

We required a speech recogniser which could transcribe continuous speech for a medium sized sub-language which was highly structured, and yet fairly flexible.

THE SPEECH RECOGNISER

Since, at the start of the project we did not know the true requirements of a speech recognition device, we chose the commercially available Speech Systems Incorporated Phonetic Engine 500 (SSI PE500)\footnote{The PE500 is available from Speech Systems, Inc. 2945 Center Green Court South, Boulder, CO 80301-2275, USA. Tel: 303.938.1110 FAX: 303.938.1874} speech recognition development kit (SDK) for the IBM Personal Computer. The PE500 aims to provide for continuous, speaker-independent speech recognition, with a 400,000-word vocabulary. The system is provided with two generic speaker models: American male and American female. The speaker model is static and hence cannot be adapted to a British speaker. Since the development of speaker models is an extensive undertaking, it must be carried out by SSI, under contract.

Words not in the vocabulary can be generated by a generalised phonetic transcription algorithm, giving an almost infinite possible lexicon. The number of active words at any one time is controlled by a context-free rewrite grammar of possible utterances. This is precompiled by the developer before use, and does not allow any adjustments to the syntax structure at run time.

We did not wish to use one of the many 'research' speech recognition systems for a number of reasons, despite their greater applicability to the problem. The foremost reason was our desire not to develop a speech recognition system tailored to our task with the large overhead that this would incur. We wanted to see how good commercial, off-the-shelf packages really are, and of course such packages are generally easier to obtain.

The PE500 is aimed at continuous speech recognition for highly structured, low perplexity, command-control applications. Whilst there is no theoretical limit to the number of active words at any one time, there is a continual degradation in performance as the size of the vocabulary and the ambiguity licensed by the syntax increases. This system is not suited for the highly perplex domain of ATC transmission, but was all we had access to at the time.

THE LBA CORPUS

The LBA Corpus was edited to facilitate the analysis of the domain language and has been manually phrase-tagged with around 50 semantic/functional labels. The creation of discourse and semantic functional phrase tags is intended to enhance the existing context-free grammar in order that it might be partitioned to take advantage of the PE500's ability to switch between applicable syntaxes. The utterances have been grouped into dialogues between the ATC and a particular pilot. The controller may be interacting with several pilots in parallel, in which case each pilot-controller 'thread' constitutes a separate dialogue. The corpus should provide evidence of habitual repeated patterns or structures within dialogues, if they exist. For example, consider the interaction between the pilot of aircraft G-AJCT and the ATC, below. The ATC's utterance ('A::') has been tagged in terms of semantic/functional labels. The number in brackets preceding the utterance is the transmission index.

(166) P: leads approach good morning golf alpha juliet charlie tango is passing 1400 feet on the heading of 240

(167) A: [CALLSIGN charlie tango CALLSIGN] [GREET leads good morning GREET] [INFO_ID you are identified INFO_ID] [MAN_HEAD continue heading two four zero MAN_HEAD]

THE TEST MATERIAL

We want to show the effect differing levels of 'linguistic knowledge' can have on speech recognition accuracy. How does the system perform with a large, perplex syntax when compared to partial information about key phrases? Is having a syntax much more accurate than simply having a lexicon? Does use of discourse greatly improve recognition? In order to eventually test different
facets of constraints, test material was chosen to reflect a number of properties. These include:

- use of one or more pieces of key-phrase information within a single utterance.
- use of aircraft identifier, otherwise known as callsign, with other key-phrase information, and with non-key information.
- discourse progression with same pilot, consisting of one complete dialogue
- at least 10 utterances.

Given the above criteria, an interaction in the corpus between the ATC and aircraft 908 was chosen, consisting of 19 utterances by the ATC (see Appendix 1).

The PE500 VoiceMatch Toolkit allows integrated collection and testing of speech material and can offer statistics on the accuracy of the decode. Six speakers were used to record the utterances using a proprietary noise-cancelling microphone. Three of the six were female. Recording occurred in a noise-controlled workspace, whilst an extra set of one speaker were recorded under normal office conditions.

The Toolkit allows the developer to use differing parameter settings when decoding speech into transcribed text. These vary by the slider setting and the language weight setting. The slider setting determines the ratio of accuracy to speed used by the decoder, i.e. how much effort the decoder puts into decoding an utterance. The PE500 has seven predetermined settings, three of which were used, approximately generating an increasing level of effort used by the decoder. The chosen slider settings were hence:

- 0, 3, 6

With each slider setting it is possible to vary the language weight, or transcription penalty value. This is a negative value which penalises excessive transcription of words, i.e. those output by the decoder. The larger the negative value, the greater the penalty and the fewer words output by the decoder. The weight needs to be optimised so that the correct number of words are transcribed. Values ranged between 0 and -150. Five values were chosen:

- 0 (default - no penalty), -40, -80, -120, -150 (maximum penalty)

MEASURES OF ACCURACY

What constitutes an accurate transcription, and how can this accuracy be graded? PE500's VoiceMatch Toolkit decodes an utterance and then attempts to align it with a template of what the utterance should actually be. This results in a number of words matching the template. Words which occur in the decoded text but not in the template are either deleted or substituted. Words which are in the template but not in the decoded text are inserted. Hence there are a number of measures which can be taken into account when calculating the accuracy of the decoded text. The following reflect those which are readily derived from the VoiceMatch Toolkit:

- number of words in input (in template)
- number of words in output (decoded text)
- number of words correct in output, occurring at appropriate place
- number of words needed to be inserted / substituted / deleted to match input

We chose a measure of accuracy based not only on the number of words correct in the output of the system, but also on the number of words actually output, i.e. transcribed. This compensates for over-generation where many more words are transcribed than occur in the speech.

WE%, the percentage of the number of words correct in the decoded text taking into account the deviation of output to input ratio.

\[
\text{number of words correct} \div (\text{number of words in input} + 1) \times (\text{number of words in output} - \text{number of words in input}) \times 100
\]

where \(|x|\) is the absolute value of \(x\).

The above measure was calculated for two scenarios: for all words in the template, regardless of whether or not they are in any of the five "key information phrases" (see Introduction) and for words which are only in one of these five phrases. The test material in Appendix 1 indicates which words fall into either category.

SYNTAX 1: BASE SYNTAX

In order to make comparisons between different syntaxes, the first set of decoding was performed using a 'base' syntax. To set the testing base, the
decoder was tested using what is equivalent to a null syntax. This gives the system no knowledge of utterance structure nor permissible utterance sequences. As required by the PE5000, the lexicon of the corpus was provided. The base syntax was simulated using an iterative word category which contained all of the words in the corpus. Thus an utterance could consist of one or more of the words in this category. The lexicon consisted of approximately 380 words.

One problem regarding the results was the inability of the system to cope with the number of words decoded from one speaker, using a default language weight of 0. The memory problem caused the system to ignore the test set. To enable further comparisons to be conducted on the results, dummy values were substituted for these results. In this case, WE% = 0.0.

SYNTAX 2: KEY-PHRASE SPOTTING SYNTAX

The second syntax we tested used the same iterative mechanism as that used in the base syntax. In effect, key-phrases were structurally defined, but could have unrestricted words surrounding and between them. In order to restrict the ambiguity of these non-key words they were limited to what occurred immediately before and after each key-phrase. The words were taken directly from the corpus. This syntax performed a kind of key-phrase spotting and allowed 'unrestricted' speech to occur in the same utterance. It is part way between the previous, lexicon-only syntax, and a full structured syntax.

Since key-phrases were to be recognised, the syntax comprised semantic/functional tags, rather than the conventional phrase structure tags. For example, the key-phrase for changing frequency was represented by a semantic tag "ALTER_FREQUENCY" which then was defined using similar tags. The whole syntax consisted of 47 "tags" or non-terminal symbols and 30 defining rules.

SYNTAX 3: FULL CONTEXT-FREE SYNTAX

The third syntax took the key-phrases of the previous, key-phrase spotting, syntax and combined them with structured non-key ('noise-phrases') so that the entire corpus could be parsed by the whole syntax. The syntax consisted of a total of 98 tags, 29 of which related to the structure of key-phrases and 55 of which related to the structure of non-key phrases. The syntax consisted of 97 defining rules.

SUMMARY OF RESULTS, ALL WORDS

Table 1 below is a summary of the recognition accuracy for the various combinations of slider settings and language weights. The combination with the best average was chosen to represent the best and worst performance for that syntax. The values shown are calculated using the WE% measure based on all words in the template. Following the table is a more detailed summary of the results for each syntax.

Base syntax

The best result was from slider setting 3, language weight -80 with an accuracy of 24.91%. The poorest result of 0% accuracy was due to aforementioned transcription problem. The next worse result was of 9.15% for slider setting 0, language weight -40. The base result taking the average for each combination of slider and language weight was 19.32% for slider 0 and weight -80. For all three slider settings, the best weight to use was -80, whilst the worst was 0. No single utterance was 100% correctly transcribed.

Key-phrase syntax

Again the best results were from using a language weight of -80, with a slider setting of 6. The best result was 26.39%. The poorest performance came from using no language weight (i.e. 0) at 7.45% for a slider setting of 0 and weight of 0. The best average result was for slider setting 6 and weight -80 at 21.67%. No single utterance was 100% correctly transcribed.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Slider</th>
<th>SSF</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>0</td>
<td>-80</td>
<td>24.65</td>
<td>13.52</td>
<td>19.32</td>
</tr>
<tr>
<td>Key-phrase</td>
<td>6</td>
<td>-80</td>
<td>26.39</td>
<td>16.56</td>
<td>21.67</td>
</tr>
<tr>
<td>Full</td>
<td>6</td>
<td>-40</td>
<td>64.48</td>
<td>47.63</td>
<td>55.26</td>
</tr>
</tbody>
</table>

Table 1: Summary of results for all words
Full syntax

The best results appeared with the use of low transcription penalties (i.e. weight of 0 and -40), at 68.06% for slider setting 6 and language weight 0. In this case, the greater the penalty, the poorer the results. The lowest was 4.09%, occurring with slider setting 0 and weight -150. The best of the averages was 58.30% with the same settings as for the best result. This setting combination also correctly transcribed a total of 15 utterances in their entirety.

SUMMARY OF RESULTS, KEY-PHRASES ONLY

Table 2 represents the same information as the previous one above. The combination with the best average was chosen to represent the best and worst performance for that syntax. The values shown are calculated using the WE% measure based on only the words which occur in the key-phrases in the template. A more detailed summary of the results for each syntax follows.

Base syntax

As can be seen, there is an insignificant improvement between the accuracy of words in key phrases, and all words in the template. The best result was an accuracy of 26.51% for slider setting 0, language weight -120. The best average result was 20.51 for slider setting 3, language weight -80. For all slider settings, best results were obtained from using language weights of -80 and -120. The poorest results can from using a low language weight, i.e. 0 or -40. No single utterance was 100% correctly transcribed.

Key-phrase syntax

Once again, the best results for each slider setting were from using language weight -80. The best results were 29.07% for slider setting 0, and on average, 22.36% for slider setting 6. The poorest results for each slider setting were from using language weight 0, at 10.04 for slider setting 3.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Slider</th>
<th>SSF</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>3</td>
<td>-80</td>
<td>25.29</td>
<td>16.84</td>
<td>20.51</td>
</tr>
<tr>
<td>Key-phrase</td>
<td>6</td>
<td>-80</td>
<td>28.09</td>
<td>17.62</td>
<td>22.36</td>
</tr>
<tr>
<td>Full</td>
<td>6</td>
<td>-40</td>
<td>73.17</td>
<td>53.89</td>
<td>64.88</td>
</tr>
</tbody>
</table>

Table 2: Summary of results for words occurring in key-phrases only

Full syntax

The best result was from slider setting 6 with language weight -40, at 73.17%. The best of the averages was 64.88% for the same settings. The language weight of -40 gives the best results for all slider settings, and once again, the larger the transcription penalty, the poorer the results. The poorest result was 11.8% using slider setting 3 and language weight -150.

COMMENTS ON RESULTS

The first syntax's use of iteration results in over-transcription of short words. This is demonstrated to its extreme by one speaker's decoded text taking more memory than the system can cope with. As the transcription penalty is increased, fewer words are transcribed and accuracy is improved. The best performance was from using large penalties, up to a certain limit. The largest imposed penalty subsequently degraded performance. There was a little improvement for key phrase words. This, however, was not considered significant.

One would expect that the second syntax would improve the accuracy, at least for the structured key phrases. There was an small increase in accuracy from the first syntax, and again a small improvement between all words and words in the key phrases. A problem with the PE500 is the inability to use any form of weighting mechanism in order to prefer key phrase words over, say non-key phrase words. This could account for the over transcription of non key phrase words in similar circumstances as the first syntax. A moderate language weight is optimal in this case.

The third syntax did not rely on the iteration mechanism, but instead consisted of definining rules. This syntax is large and ambiguous but greatly improved recognition. Once again, there is a small increase in performance for those words in the key-phrases. Most surprisingly, however, the best results come from using either no transcription penalty or the smallest. This could reflect the PE500's inability to accurately transcribe syntaxes which make extensive use of the iteration mechanism.
The first two syntaxes show that there is little difference between one's choice of slider setting, whereas the third syntax shows the opposite with large differences in performance. Use of the iteration mechanism results in over-transcription, hence requiring a higher transcription rate penalty for better results. This is not the case for the third syntax which gives better results for a low transcription penalty values.

**USING HIGHER LINGUISTIC LEVELS: TOWARDS A GRAMMAR OF DISCOURSE**

We wish to see the effect that higher levels of linguistic information have on the speech recognition performance. In particular, we would like to explore the effect of using a discourse grammar on what is intuitively a well-structured domain. A large, all-encompassing syntax, such as syntax 3, can be broken down into smaller, well-defined subsets provided that there is a definite distinction between dialogue segments in the domain. This smaller syntax is potentially less ambiguous than the original, containing fewer words and less complicated structures. If this is the case, one would expect that the application of this smaller syntax to result in a higher recognition rate.

To obtain some initial results for such use of a syntax, a further set of experiments were conducted using a single subset of syntax 3. This syntax contained enough information to cover the entirety of the test material. Although the combination of key-phrases was reduced, the full expressiveness of the phrases were preserved. For example, although the new syntax would not allow a callsign followed by a change of frequency, it would allow a callsign followed by a change of heading. The choice of callsign is from the original universe of callsigns and the headings still reflect all of the possible changes in heading.

The revised syntax contained 50 tags, one of which defined the start of the utterance, and 48 rules or word categories. The lexicon consisted of 257 words and the number of sentences which could be produced is comparable with the original syntax (compare with the original: 98 tags, 97 rules and 380 words in lexicon).

Tables 3 and 4 below summarise the results for all words in the test material and for key-phrase words only.

For all words, the best performance of 75.53% came from using a slider setting of 6 and language weight of -40. The trend in results is very similar to those for the full syntax where a greater transcription penalty leads to poorer results. The best average was 66.33% with a slider setting of 6 and no transcription penalty. This is 8.03% higher than the respective original syntax. This combination of slider and penalty gives a total of 26 sentences transcribed without any errors, 11 more than the original syntax.

The best result of 78.92% came from a combination of a slider setting of 6 and no language weight. The best average of 71.28% was obtained from the same settings. This is an increase of 6.4% on the original syntax.

It is not surprising to see the same trends in this syntax as in the original. A low or non-existent language weight gives the best results. An increase of around 8% may not be much but does highlight the increase in performance by using smaller subsets. The subset used in this case was comparable to the original since it was still a large and potentially ambiguous syntax. We hope that the use of smaller subsets, applied through a discourse grammar would lead to greater improvements in performance.

<table>
<thead>
<tr>
<th>Slider</th>
<th>SSF</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
<th>No. Utts Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0</td>
<td>74.18</td>
<td>59.66</td>
<td>66.33</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>-40</td>
<td>75.53</td>
<td>52.75</td>
<td>60.83</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 3: Summary of results for all words using subset syntax

<table>
<thead>
<tr>
<th>Slider</th>
<th>SSF</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0</td>
<td>78.92</td>
<td>63.31</td>
<td>71.28</td>
</tr>
<tr>
<td>6</td>
<td>-40</td>
<td>77.3</td>
<td>67.07</td>
<td>70.89</td>
</tr>
</tbody>
</table>

Table 4: Summary of results for words occurring in key-phrases only, using subset syntax
THE SEGMENTATION OF DIALOGUE

Discourse can be broken into discourse segments which reflect a set of utterances with some properties in common. A discourse segment can be the utterances discussing a certain topic. It can also be the discourse between a set of speakers, in other words, a dialogue. In the ATC application it is helpful to divide the total set of utterances by the ATC and respective pilots into dialogues. For example, a dialogue can be all the utterances by the ATC and pilots between the ATC starting his/her shift and finishing. A dialogue will then be all the utterances concerning the ATC and a particular pilot. Individual dialogues can be further divided into segments indicating the flow of the discourse.

For this approach to work, we need a method for dividing the dialogue into maximally distinct discourse segments. Unfortunately, discourse grammar is a loosely-formalised area with few formal guiding principles, so we turn to automatic "Machine Learning" techniques for segmentation. Corpus-based statistical clustering techniques have been applied to other segmentation/labelling problems in NLP, e.g. clustering words into word-classes [Atwell & Drakes 87, Hughes 94, Hughes & Atwell 94], and clustering texts into related languages [Churcher 94, Souter et al. 94].

The automatic segmentation of a dialogue should provide the basis for the generation of a discourse grammar. A discourse grammar would allow a speech recognition system to apply syntaxes which have immediate relevance to the utterances being spoken at the time. Furthermore, additional language models can be applied to the discourse structure as it evolves.

DIALOGUE SEGMENTS

The ATC Approach corpus is already divided into utterances between a pilot and the ATC. Each set can be thought of as a discourse segment.

One feature of the ATC dialogues is that they can be interleaved with one another, posing the problem of dialogue tracking. This has partially been tackled in [Grosz 86] and other modelling strategies.

As an example, a dialogue can be split up into functional units: a segment can be thought of as a GREETING exchange, some INFORMATION exchange and a SIGNING OFF exchange, where a protocol for ending the dialogue exists. With other discourse segments, each of these units may consist of more utterances or fewer, or introduce other, finer units.

METHOD OF SEGMENTATION USED

In order to assist the generation of a discourse grammar, it is useful to look at the semantic labels used throughout the corpus. Here is an example dialogue extracted from the corpus. Only the semantic tags are shown for clarity:

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1 | (34) | [+CALL] | [GREET] |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 2 | (38) | [+CALL] | [AFFIRM] | [INFO_CURRENT] | [+INF_QNH] |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 3 | (38) | [+CALL] | [AFFIRM] |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 4 | (39) | [+CALL] | [REQ_CONFIRM] |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5 | (41) | [+CALL] | [THANKS] | [INFO_POS] | [INFO_END] | [+ALT_FR] | [INFO_LOC] |   |   |   |   |   |   |   |   |   |   |   |
| 6 | (43) | [BYE] |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

The simplest method of automatically dividing the discourse is to divide it into roughly equal parts based on the number of sub-segments desired. For example, two 'clusters' would divide the discourse into utterances 34-38, 39-43. Three 'clusters' would divide it into 34-36, 38-39, 41-43.

Taking each set of clusters for all discourse segments, the similarity between different sub-segments can be calculated using some measure. We decided to initially try our approach using the key information phrase labels only, ignoring the noise information.

COMMENTS ON CLUSTERING APPROACH CHosen

The above segmentation technique is very simple and thus suffers from a number of disadvantages. As can be seen from the example, choosing three or less clusters will result in the incorrect placing of utterance 36 into the first sub-segment.
Dividing the segment by hand into functional units resulted with utterance 36 being placed into sub-segment 2, i.e. the INFORMATION exchange unit. The strict division of dialogue into 'roughly' equal parts results in utterances being placed into wrong sub-segments.

One way to view the discourse segment is as a continuum of semantic tags, both because of the above problem and due to the more or less uniform distribution of some common sequences of tags. A technique which can be adapted for this purpose is explained in [Hughes 94]. Hughes uses a normalised frequency distribution of word / word-type position within a sentence. For example, consider the frequency distributions in figure 1 for three tag sequences.

Figure 1: (Y: frequency of rule; X: utterance position in segment)

The example tag sequences show the following:

(a) a definite peak towards start of discourse segment
(b) a definite peak towards end of discourse segment
(c) no definite peak - a more or less uniform distribution throughout discourse segment

Frequency distributions and hence derived probability distributions can be used by the discourse level instead of using distinct segments to distinguish between differing sections of discourse. This approach combats the problem of utterances which are divided into the incorrect segment.

MEASURE OF SIMILARITY BETWEEN SUB-SEGMENTS

A bigram frequency model was generated for each cluster set. This simple model of sequences of tags in clusters allowed a correlation coefficient to be calculated and clusters within the same set compared.

First, the corpus of dialogues was divided according to the number of clusters chosen, then given to an n-gram model generation program. The statistical package, SPSS was used to generate the correlation coefficient between different pairs of clusters. This data was then used by a clustering package to generate dendograms indicating the similarities between the clusters. The clustering algorithm used was Ward's which uses a statistically based dissimilarity measure [Ward 63, Wishart 69] favoured by Hughes [Hughes 94] for clustering words.

CLUSTERING RESULTS

Four sets of clusters were generated, using clusters of number 3, 4, 5 and 6. The dendograms of three and five clusters in figures 2 and 3 below show the grouping of different clusters, the closer to the right a join between two clusters, then the greater the similarity between them.
CONCLUSIONS FROM INITIAL CLUSTERING METHOD

The correlation values showed that many of the clusters were very similar. The greater number of clusters chosen, the greater the variance between them. At five clusters, the correlation coefficient between the first and the last cluster drops to 0.7542, the lowest value present.

Another approach which should be considered is that of an intention or plan level, one level higher than the discourse level. Just as syntax is considered as parts of discourse segments, discourse can be considered as parts of a plan. For example, the frequency distribution of tags in one discourse segment where a pilot intends to land at the airport may be quite different to that of one where the pilot is taking off and leaving the ATC area. This difference in the plan or intention of the pilot should be taken into consideration when segmenting the discourse.

Dividing dialogues into sets which have the same intention / plan generates a problem of its own. A much greater number of segments are required, and hence a larger corpus, in order to provide adequate numbers of instances.

There has to be evidence that each discourse sub-segment is distinct enough from its neighbours in order to create a discourse grammar which is more effective than simply using a single syntax, [Churcher et al. 95]. Initial correlation coefficients show that there is little difference between successive sub-segments. However, this may be the result of using a very simple and error-prone clustering method. Further work using a dynamic clustering method or frequency distributions should be considered before concluding that a discourse grammar is unfeasible in this instance.

USE OF CONTEXTUAL INFORMATION

The use of a natural language component to constrain the output of the system could increase the system’s recognition performance. In this domain, there is also a wide range of contextual knowledge which could be incorporated into the system, either by means of a database containing information applicable to the local area around the ATC, or by controlling the speech recognition unit itself. The contextual knowledge which could be applicable includes the following:

1. Current callsigns being used in airspace.
2. Current transponder settings (squawks) being used by aircraft.
3. Current pressure settings of the local area, etc.
4. Regional geographical landmarks.
5. Transponder code ranges used at LBA.
6. Radio frequencies used at or around LBA.
7. Runway identifiers used at LBA.

The first three items contain information which exists for differing periods of time. For example, the callsigns currently being used exist only for the duration that the pilot is in LBA airspace. The remainder of the information is local to LBA, itself.

As an example of how this information may be used, consider the transponder or 'squawk' codes which range in value from 0400 to 0420, in octal and that only one aircraft in LBA airspace can have a
CONCLUSIONS FROM INITIAL CLUSTERING METHOD

The correlation values showed that many of the clusters were very similar. The greater number of clusters chosen, the greater the variance between them. At five clusters, the correlation coefficient between the first and the last cluster drops to 0.7542, the lowest value present.

Another approach which should be considered is that of an intention or plan level, one level higher than the discourse level. Just as syntax is considered as parts of discourse segments, discourse can be considered as parts of a plan. For example, the frequency distribution of tags in one discourse segment where a pilot intends to land at the airport may be quite different to that of one where the pilot is taking off and leaving the ATC area. This difference in the plan or intention of the pilot should be taken into consideration when segmenting the discourse.

Dividing dialogues into sets which have the same intention / plan generates a problem of its own. A much greater number of segments are required, and hence a larger corpus, in order to provide adequate numbers of instances.

There has to be evidence that each discourse sub-segment is distinct enough from its neighbours in order to create a discourse grammar which is more effective than simply using a single syntax, [Churcher et al. 95]. Initial correlation coefficients show that there is little difference between successive sub-segments. However, this may be the result of using a very simple and error-prone clustering method. Further work using a dynamic clustering method or frequency distributions should be considered before concluding that a discourse grammar is unfeasible in this instance.

USE OF CONTEXTUAL INFORMATION

The use of a natural language component to constrain the output of the system could increase the system's recognition performance. In this domain, there is also a wide range of contextual knowledge which could be incorporated into the system, either by means of a database containing information applicable to the local area around the ATC, or by controlling the speech recognition unit itself. The contextual knowledge which could be applicable includes the following:

1. Current callsigns being used in airspace.
2. Current transponder settings (squawks) being used by aircraft.
3. Current pressure settings of the local area, etc.
4. Regional geographical landmarks.
5. Transponder code ranges used at LBA.
6. Radio frequencies used at or around LBA.
7. Runway identifiers used at LBA.

The first three items contain information which exists for differing periods of time. For example, the callsigns currently being used exist only for the duration that the pilot is in LBA airspace. The remainder of the information is local to LBA, itself.

As an example of how this information may be used, consider the transponder or 'squawk' codes which range in value from 0400 to 0420, in octal and that only one aircraft in LBA airspace can have a
particular code. This information can assist the choice of the correct code.

CONCLUDING REMARKS

The above results show the advantages of using a full, context-free syntax in the domain of Air Traffic Control transmissions using the formalism provided by the PE500. The use of key-phrase spotting with the mechanism of iteration produced inaccurate transcriptions with results little better than not having a syntax at all. Some form of weighting mechanism for the key-phrases may be of value in increasing the performance.

The PE500 is designed for low vocabulary, low perplexity, command-control speech recognition. It is not designed to perform well on large and ambiguous syntaxes and this is reflected by the results. Its performance is poor when compared to the research systems used in the recent ARPA Wall Street Journal competition [Collingham 94, ARPA 94] but it must be noted that the system was not "trained" nor optimised for the domain or speakers, except that a syntax was provided. Hence, this set of experiments have been a comparative study of the use of differing levels of linguistic information using a commercially available speech recogniser.

The use of a discourse grammar to divide the large syntax into smaller syntaxes may improve performance. The smaller syntaxes may perform better due to lower perplexity and ambiguity and could be applied as the discourse progresses. Such use of higher level "linguistic knowledge" together with contextual information should, in theory, improve the performance of the continuous speech recogniser. The representation of such a discourse grammar is not clear. Automatic clustering of a corpus may assist the identification and representation of distinct dialogue segments, if they exist for a particular domain language.

BIBLIOGRAPHY


[Hughes 94] J Hughes. “Automatically Acquiring a Classification of Words”. Ph.D.


APPENDIX 1

TEST 908 SENTENCE LIST (KEY SUB-PHRASES ARE UNDERLINED)

1. nine zero eight standby for further descent expect vector approach runway three two information charlie current q n h one one zero five and q f e nine nine one millibars
2. nine zero eight report your heading
3. nine zero eight roger continue that heading descend to altitude four thousand feet leeds q n h one zero one five
4. flight knightair nine zero eight turn left heading zero eight five
5. two eight nine zero eight leeds
6. runway one four is available vectors to a visual approach if you wish give you about two seven track miles to touchdown
7. expect a visual approach runway one four q f e nine nine zero millibars proceed descent altitude three thousand five hundred feet
8. q f e nine nine zero millibars for runway one four
9. two eight nine zero eight turn right heading one zero zero
10. nine zero eight roger maintain
11. two eight nine zero eight descend to height two thousand three hundred feet q f e nine nine zero millibars
12. on that heading you'll be closing for a visual final that's about five miles you've got approximately one one track miles to touch down
13. nine zero eight descend height one thousand five hundred feet q f e nine nine zero
14. nine zero eight your position five north west of the field report as you get the field in sight
15. zero eight nine zero eight turn right heading one four zero
16. zero eight nine zero eight descend to height one thousand two hundred feet
17. on the centre line three and one half miles to touchdown
18. thanks happy to continue visual
19. contact the tower one two zero decimal three
Context Construction as Subtask of Dialogue Processing – the VERBMobil Case

Elisabeth Maier*
DFKI GmbH
Stuhl satzenhausweg 3
D-66123 Saarbrücken
maier@dfki.uni-sb.de

Abstract

This paper presents the dialogue component of the speech translation system VERBMobil. In particular, it describes the Dialogue Memory which has been developed to represent contextual information acquired during dialogue processing. Information is stored both chronologically, i.e. in the order of appearance (in the Sequence Memory) and conceptually (in the Thematic Structure). We show how the Dialogue Memory is used to draw contextual inferences, some of which serve as basis for the detection of indirect speech acts.

1 Introduction

This paper presents an approach for the representation of contextual information that was implemented as part of the face-to-face speech translation system VERBMobil. The content of the so-called Dialogue Memory in which contextual information about the ongoing dialogue is stored has been very much determined by the requirements of a number of system components, like e.g. speech recognition, semantic processing, transfer and generation (a more detailed discussion of this issue can be found in [11]). In VERBMobil, it is a task of the dialogue component to incrementally construct a representation of the context.

This paper first gives a brief presentation of the dialogue component and the subtasks it has to fulfill in VERBMobil (section 2). After a discussion of the two basic knowledge types that are represented in our context model – dialogue acts (section 3.1) and time information (section 3.2) – we show the two submodules of which the Dialogue Memory consists: Sequence Memory and Thematic Structure (section 3.3). To illustrate the incremental construction of the context representation in the Dialogue Memory we give an example for the processing of a dialogue fragment from our corpus of appointment scheduling dialogues (section 3.4). Finally we describe how contextual inferences are supported by our Dialogue Memory (section 4) and indicate how this approach can also be used to handle indirect speech acts (section 5).

2 Dialogue Processing in VERBMobil

It is a key issue of our project to establish robust processing methods that can cope with unreliable and incomplete input as it is typical for spoken language systems. One means to achieve this is the availability of contextual information. In VERBMobil this information is used, e.g. to predict follow-up speech acts in dialogue processing, to disambiguate translational equivalents during transfer, to resolve anaphoric expressions in semantic evaluation and to control lexical variation in the generation of target language expressions. In our case, contextual information is provided by the dialogue component [2]. It is the task of this component to monitor the progress of the dialogue and to provide a representation of what has been said. On this basis the dialogue component is able to constrain decisions made by other system components and to predict follow-up dialogue states.

*This work was funded by the German Federal Ministry for Research and Technology (BMBF) in the framework of the VERBMobil Project under Grant 01IV101K/1. The responsibility for the contents of this study lies with the author. The author wishes to thank Jan Alexanderson, Norbert Reithinger, and Adelheit Stein for comments on earlier drafts of this paper, and Martin Kleen for his help in implementing the Dialogue Memory.
The dialogue component has been realized as a hybrid architecture: it contains statistical and knowledge-based methods. Both parts work with dialogue acts [3] as basic units of processing.

The statistics module is based on data automatically derived from a corpus of dialogues that have been manually labeled with dialogue acts. On the basis of this knowledge the statistics module determines possible follow-up dialogue acts for every utterance (see [14]). The plan recognizer as knowledge-based module of the dialogue component incorporates a dialogue model, which describes sequences of dialogue acts as occurring in appointment scheduling dialogues (see [1]).

3 Context Construction

As explained in [11] we took a rather “pragmatic” approach to the design of the Dialogue Memory: we decided to include only information types which are required by subcomponents of VERMMOBIL. Therefore we took the needs of speech recognition, syntactic-semantic processing, transfer and generation into account. In the following, we discuss the two most prominent information types included in our Dialogue Memory, namely dialogue act and temporal information. We explain how these information types are represented in two subcomponents of the Dialogue Memory, the Sequence Memory and the Thematic Structure. We conclude this section by providing an example for the construction of the context representation when processing a sample dialogue.

3.1 Dialogue Acts for Appointment Scheduling Dialogues

On the basis of an extensive corpus of appointment scheduling dialogues (currently about 1000 dialogues are available in transliterated form) we determined a set of 42 dialogue acts [8]. These dialogue acts form the leaves of a dialogue act hierarchy, where the more abstract levels are rather independent from the domain at hand. In figure 1 we show only the abstract dialogue act categories; domain-dependent dialogue acts concern e.g. dates, locations and durations. The hierarchical organization of the dialogue acts has a number of advantages, one of them is the applicability of acts to other domains and applications. We expect, for example, that the set of abstract dialogue acts is easily portable to other negotiation dialogues; first experiments with travel planning dialogues confirmed this assumption.

The hierarchy given in figure 1 shows that three phases can be distinguished in appointment scheduling dialogues: an initialization phase (e.g. GREET, INTRODUCE), a negotiation phase (e.g. SUGGEST, ACCEPT) and a closing phase (e.g. CONFIRM, THANK). While the initialization and the closing phase fulfill rather social functions, the negotiation phase is mainly task-oriented: dialogue acts belonging to this phase are used to advance the negotiation and to achieve the goal of the interaction. Another class of acts, the deviations, cannot be properly attributed to one dialogue phase: they can occur at any point of a dialogue. They also do not contribute to the task as such; they are used to describe actions like e.g. giving feedback, thinking aloud, etc.

In this paper we focus only on the dialogue acts of the negotiation phase; they either co-occur with propositional material relevant to the task, i.e. information about proposed dates and times, or they present an evaluation of proposals (i.e. acceptance or rejection).

3.2 Time Information in Appointment Scheduling Dialogues

In addition to dialogue acts another information type that is relevant for the advancement of appointment scheduling dialogues is temporal information. To represent temporal information in our Dialogue Memory we use an intuitive hierarchical model of temporal categories. Our categories are date, year, month, week, etc. down to time as most fine-grained type of temporal information. These categories are ordered hierarchically insofar as an instance of a temporal category is refined by an instance of a category of finer granularity.
i.e. an instance of type year can be specified further by adding an instance of type month. In this case we say that the instance of type year is superordinated to the instance of type month. For every temporal information mentioned during the conversation one or more instances of the appropriate categories are created and embedded into the temporal structure created so far.

3.3 Sequence Memory and Thematic Structure

In the Dialogue Memory of the VERBMOBIL system two subcomponents have been developed for the representation of context: the sequence memory, which mirrors the sequential order in which the utterances and the related dialogue acts occur, and the thematic structure\(^1\), which consists of instances of temporal categories and their status in the dialogue, i.e. who they have been proposed by and whether they have already been accepted or rejected. Both components are closely intertwined so that for every utterance of the dialogue the available information can be easily accessed.

During dialogue processing contextual information is constructed as follows (see figure 2):

- **Updating the Sequence Memory**
  - **Determination of Dialogue Act**
    the dialogue act included in an utterance is determined either through shallow or deep processing; in the latter case the semantic evaluation component is responsible for the computation of the dialogue act. Depending on the quality of the translation either the results of shallow processing or of semantic evaluation are incorporated into the Dialogue Memory. While shallow processing computes dialogue acts mostly on the basis of key words the semantic evaluation component additionally uses sentence mood and contextual information. After dialogue act determination the Sequence Memory is updated by an object which includes the dialogue act together with the identifier of the utterance. This new object is then linked to the object representing the previous utterance.
  - **Computation of Predictions**
    on the basis of the dialogue act the statis-

tical component computes the most likely dialogue acts for the following utterance; this information is then entered into the representation of the current Sequence Memory object;

- **Plan Recognition**
  using the dialogue act the plan recognition component determines how the current utterance fits into the expected course of an appointment negotiation dialogue. It also determines which dialogue phase the utterance belongs to; this information is then added to the Sequence Memory.

- **Parsing of Time Expressions**
  using the information supplied by the parser the semantic evaluation component retrieves the temporal information included in an utterance\(^2\) and maps it into an expression of a time description language which was developed specifically for the purposes of VERBMOBIL. This language (for details see [9]) is designed to model temporal knowledge that can be directly entered into the Thematic Structure (absolute temporal information), and temporal information that requires additional inferences in order to be added to the Thematic Structure (relative temporal information)\(^3\). Some of the relative expressions, in particular those expressions representing temporal references (e.g. *then, that day*), are resolved by the semantic evaluation component on the basis of focus information provided by the Thematic Structure. The majority of the relative expressions, though, requires additional processing in order to be expressible in terms of the Thematic Structure.

- **Computation of Absolute Time Expressions**
  relative expressions are, for instance, referring expressions (e.g. *last Sunday, next week*), temporal modifiers (e.g. *early June, late afternoon*), public holidays (e.g. *Christmas, Good Friday*) and combinations of these categories (e.g. *Thursday in three weeks, the day after Easter*); for an outline of this process see [7]). Some of

---

\(^1\) The term *thematic structure* is unrelated to the same term as introduced in the Prague School of Linguistics. The term rather refers to what the dialogue is about, i.e. its propositional content.

\(^2\) For the time being shallow methods for the extraction of time processing are not integrated in our prototype. It is foreseen for later stages of the project, though.

\(^3\) A similar distinction for temporal expressions can be found in [12].
these computations, in particular the resolution of referring expressions, are made on the basis of a so-called reference point or focus which is the time point under consideration at the current stage of the dialogue. The reference point is updated as the dialogue proceeds and the participants' attention moves on to other time points.

- **Embedding in Thematic Structure**
  with the temporal information once mapped onto the appropriate categories the Thematic Structure is either updated with newly created instances in case the time frames have not been mentioned yet or the focus is shifted back to objects already mentioned and therefore available in the Thematic Structure.

- **Inferencing over Thematic Structure**
  under conditions that will be discussed in section 4 the entry of a new temporal instance into the Thematic Structure induces a number of follow-up actions, like e.g. the propagation of dialogue act or evaluation information to subordinated or superordinated instances.

### 3.4 A Running Example

In this subsection we show how a part of a sample dialogue is processed and subsequently represented in the Dialogue Memory. The dialogue together with the dialogue acts and time expressions for the individual utterances, i.e. the output of Semantic Evaluation, are given in figure 3.

While processing the dialogue the system creates an object for every turn, i.e. for every contribution made by one speaker and adds it to the Sequence Memory. The information concerning the individual utterances is then linked to this turn, mirroring their order of occurrence (see figure 4 for a snapshot of the Sequence Memory after processing the sample dialogue).

In the Thematic Structure the first proposal made by dialogue partner A leads to the creation of time objects for the month, day, weekday and the period-of-day. These objects also represent information concerning the speaker who made the suggestion. In the next utterance the proposal is rejected by speaker B. For doing so, an anaphoric expression is used. Since anaphoric expressions by default refer to the temporal objects which are currently in focus, i.e. which have been mentioned in the previous utterance, the dialogue act of the current utterance refers to those objects. Because for anaphoric expressions rejections only refer to the most specific object in focus (rejecting a Thursday morning as time for a meeting does not necessarily mean that all of Thursday is rejected) only the object concerning the period of day is updated with the corresponding information. The full Thematic Structure for the dialogue segment is given in figure 5.

### 4 Contextual Inferences

In this section we have a closer look at the construction of the Thematic Structure: it can be observed that dialogue acts which concern a specific time object, i.e. the rejection or acceptance of such an object, can have consequences for other related time objects: they can be implicitly accepted or rejected. In the following subsections we discuss two different cases: inferences induced by **individual dialogue acts** (subsection 4.1) and inferences initialized by **chains of dialogue acts** (subsection 4.2).

To this end we introduce a more formal account for dialogue acts, for the Thematic Structure and for the operations manipulating it.
4.1 Inferences Induced by Processing Individual Dialogue Acts

When dialogue acts are related to individual time objects of the Thematic Structure, certain follow-up actions for the manipulation of related time objects are induced. An example for such inferences can be observed in the following example:

GBP002: I guess we could meet on the tenth
(SUGGEST_SUPPORT_DATE)
I am free in the afternoon
(SUGGEST_SUPPORT_DATE)

SKH003: I am <āh> taking the tenth off
(REJECT_DATE)
<āh> I have the <āh> afternoon of the
eleventh available
(SUGGEST_SUPPORT_DATE) ...

AKK001: well <āhm> the day that I am free the
most is probably Friday the twentieth
(SUGGEST_SUPPORT_DATE)
I am free all day
(SUGGEST_SUPPORT_DATE)
is <āh> that okay with you
(REQUEST_COMMENT_DATE)

JEB002: I have a meeting that afternoon
(SUGGEST_EXCLUDE_DATE)
but <āh> the morning would be great
<āh> around nine or ten o'clock
(SUGGEST_SUPPORT_DATE)
we can go 'till around two
(SUGGEST_SUPPORT_DATE)

AKK003: nine o'clock sounds pretty good
(ACCEPT_DATE)

The acceptance of the time object representing
nine o'clock implies the acceptance of all superordinated
independent time objects, like the objects for morning,
Friday, twentieth, etc.

Downward and upward inheritance can be captured formally as follows:

Downward Inheritance
(REJECT_DATE x A) ⇒ ∀y (superordinated x y) | (REJECT_DATE y A)

Upwards Inheritance
(ACCEPT_DATE x A) ⇒ ∀y (superordinated y x) | (ACCEPT_DATE y A)

where (REJECT_DATE x A) and (ACCEPT_DATE x A) stand for the time object x being either rejected or accepted by speaker A, and (superordinated x y) stands for the fact that in the Thematic Structure the time object y is superordinated to the time object x.
SUGGEST_SUPPORT_DATE AB
[mont: May, day: 26, day.of.week: Thu,
period.of.day: morning]
A003: ja, das erste Treffen würd ich gerne um
Mai mit Ihnen machen, und da explizit am,
<ah> ja, am sechsundzwanzigsten Mai,
Donnerstag vormittags.
yes, the first meeting I'd like to make with you in
May, and then explicitly on the twenty-sixth,
Thursday morning.
REJECT_DATE BA
[aha]
B001: <ah> das paßt mir nicht gut.
that doesn't suit me.
REJECT_DATE BA
[after{[day: 26]}]
ab sechsundzwanzigsten <hm> kann ich
I can't starting with the 26th.
SUGGEST_SUPPORT_DATE BA
[before{[day: 26, aha]}]
vielleicht die Woche davor,
perhaps the week before (that),
REQUEST_COMMENT_DATE BA
[aha]
paßt Ihnen da irgend etwas?
does anything suit you then?
SUGGEST_SUPPORT_DATE AB
[day.of.week: Tue, day: 17]
A002: ja, da können wir am Dienstag, den
siebzehnten, vielleicht ?
yes. Then we could on Tuesday the seventeenth,
perhaps?
ACCEPT_DATE BA
[aha]
B003: das geht in Ordnung für mich.
that's ok with me.
SUGGEST_SUPPORT_DATE AB
[day.of.week: Tue, day: 17, period.of.day: afternoon]
A004: da machen wir's doch Dienstag,
siebzehnten, am Nachmittag,
then let's make it Tuesday, seventeenth,
in the afternoon.

Figure 3: A sample dialogue

Figure 4: The Sequence Memory after processing
the sample dialogue.
Other dialogue acts of our set that induce inheritance in the same way are suggest_exlude_date which is used for mentioning time frames that are not available for an appointment (downward inheritance), suggest_support_date, that is employed when a time is proposed for a meeting (upward inheritance), and confirm which is used to wrap up an agreed upon date (upward inheritance).

4.2 Inferences Induced by Processing Dialogue Act Sequences

In appointment scheduling dialogues we very often find that the evaluation of the content, i.e. of certain time frames, remains implicit and has to be inferred from the surrounding context. Such inferences occur frequently when a speaker change takes place. In this section we show how such inferences can be made exploiting the dialogue act history and the information represented in the Thematic Structure. In the following sections we focus on the discussion of inferences made on the basis of utterance pairs.

Counterproposal with Implicit Rejection

In our corpus of appointment scheduling dialogues we can find cases, where a dialogue partner makes a new proposal thereby implicitly rejecting a proposal made by the respective other dialogue partner. As can be seen from the following example, this information can be inferred from the propositional content of the utterance, i.e. from the information stored in the Thematic Structure.

JAK010: 'n Termin würde mir ganz gut passen, Montag, der achte, bis Freitag, der zwölfte, da hört' ich dann irgendwann Zeit. There is one date that would suit me well, Monday, the eighth, until Friday, the twelfth, around that time I would be available. suggest_support_date

REK011: ja ich denke an einem Wochenende, yes I think on a weekend suggest_support_date weil 's ja wohl andene 'n biechen länger wird. because in the evenings it will get a bit late. give_reason ähnlich zwanzigster, einundzwanzigster. twelvieth, twentyfirst. suggest_support_date

The proposal to meet around the twelfth which has been made in the last utterance of turn JAK010 is followed by a proposal made by speaker REK to meet around the twentieth. Since the two proposals are not compatible, it can be inferred that the latter proposal implicitly serves as rejection of the former. In general, it can be observed that the suggestion of a time object followed by the suggestion of an incompatible time object of the same type serves as implicit rejection. This principle can be described as follows:

\[
\begin{align*}
\text{Chained Proposals} & = \text{Implicit Rejection} \\
(suggest\_support\_date_n \: x \: A) \land \\
(suggest\_support\_date_{n+1} \: y \: B) \land \\
(type \: x) & = (type \: y) \land \\
(x \neq y) & \Rightarrow \\
(reject\_date_{n+1} \: x \: B)
\end{align*}
\]

with the indices after the speech acts indicating the number of the utterance and with type standing for the category of the time object. In short this formula means that the time object proposed by A in the nth utterance can be considered rejected by B in the (n+1)th utterance if speaker B proposed an incompatible time object in utterance (n+1).

New Proposal with Implicit Acceptance of Previous Proposal

Suggestions followed by a different proposal made by the other dialogue partner do not necessarily imply a rejection of the initial proposal – the contrary can be the case:

SRH003: ...how 'bout any time in the afternoon (suggest_support_date)

DTL004: how does three o'clock look (suggest_support_date)

The time object presented in turn DTL004 is a refinement of the proposal made in the preceding turn SRH003, thereby implicitly accepting the proposal. This principle can be described as:

\[
\begin{align*}
\text{Chained Proposals} & = \text{Implicit Acceptance} \\
(suggest\_support\_date_n \: x \: A) \land \\
(suggest\_support\_date_{n+1} \: y \: B) \land \\
(\text{superordinate} \: x \: y) & \Rightarrow \\
(accept\_date_{n+1} \: x \: B)
\end{align*}
\]
Acceptance by Downward Inheritance

As shown in the previous subsection, the acceptance of an individual time object induces the acceptance of all superordinate time objects by means of upward inheritance. Here we show that also downward inheritance may be triggered by the acceptance of a time object, in case subordinate time objects are available in the Thematic Structure. The following dialogue fragment presents an example for such a behavior:

SKH003: ... <â€” have the <â€” afternoon of the eleventh available <â€” fourteenth <â€” maybe morning of the sixteenth
(SUGGEST_SUPPORT_DATE)

GBP004: I am free all day the eleventh
(ACCEPT_DATE)

Among time objects introduced and proposed in turn SKH003 are the time objects for the afternoon of the 11th. This turn is followed by an utterance where the other speaker accepts the day. Because being available the whole day also implies having the (proposed) afternoon free, the acceptance information can be inherited down to the object of type period-of-day. In this case, therefore both upwards and downwards inheritance as introduced in section 4.1 and downwards inheritance take place. The downward inheritance of acceptance information is captured by the following rule:

Acceptance by Downward Inheritance

(SUGGEST_SUPPORT_DATE_n x A) \land
(ACCEPT_DATE_{n+1} y B) \land
(superordinate y z) \Rightarrow
(ACCEPT_DATE_{n+1} z B)

Implicit Acceptance through Phase Change

In some cases a date proposed by one speaker can be considered as implicitly accepted if the respective other dialogue participant moves on to another dialogue phase. An example for such a phenomenon can be observed in the following dialogue fragment:

JDH004: ... well I get out of class at one, DELIBERATE_EXPLICIT
so give me about a half hour or so and maybe we can get together for one thirty in the afternoon then

SUGGEST_SUPPORT_DATE
I will see you then
SUGGEST_SUPPORT_DATE
SMA005: do you want to meet in my office or mine
<â€” yours
SUGGEST_SUPPORT_LOCATION

In this example the proposed date is implicitly accepted by speaker SMA since he switches to the phase where the location for a meeting is being negotiated. Similarly, progression to the closing phase implies an acceptance of everything the dialogue partners proposed before, assuming that the two interactants show cooperative behavior.

A formal account for the implicit acceptance of a proposal through phase change looks as follows:

<table>
<thead>
<tr>
<th>Phase Change = Implicit Acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(SUGGEST_SUPPORT_DATE_n x A) \land</td>
</tr>
<tr>
<td>(&lt;dialogue.act &gt;_{n+1} y B) \in</td>
</tr>
<tr>
<td>{NEGOTIATE_LOCATION, CLOSING} \Rightarrow</td>
</tr>
<tr>
<td>(ACCEPT_DATE_{n+1} z B)</td>
</tr>
</tbody>
</table>

Of course, the principles discussed above can be freely combined in order to process also highly complex examples from our corpus of appointment scheduling dialogues. Whether there is a specific order in which these rules have to be applied and whether it is necessary to develop resolution strategies for conflicting rule applications is subject to future research.

5 The Treatment of Indirect Speech Acts - Discussion

A difficult problem in the processing of natural language utterances is the determination of indirect speech acts (for a detailed account of indirect speech acts see [15]). These speech acts occur in cases where the surface form of an utterance does not - or at least not fully - coincide with the intention that stands behind that utterance. An utterance can fulfill one or more purposes, one being clearly expressed in the surface form and the other being left to the hearers' inferences.

---

4For the treatment of some of our appointment scheduling dialogues it is necessary to divide the negotiation phase into two subphases: NEGOTIATE_DATE where a date for a meeting is being negotiated, and NEGOTIATE_LOCATION where a location for a meeting has to be determined. The negotiation of a location is not part of the official VERB-MOBIL scenario, though.
Systems that rely only on surface cues for the determination of dialogue acts are not able to capture indirect speech or dialogue acts. Keyword spotting techniques, for example, cannot be used to find indirect speech acts unless they are combined with mechanisms like focus tracking. Therefore, deep methods are required that abstract away from the surface form and that determine indirect speech acts on the basis of semantic and pragmatic evidence.

The approach proposed in the previous chapter, i.e. the incremental construction of the Thematic Structure and the derivation of additional information can be seen as an approach for the detection of a subset of indirect dialogue acts in appointment scheduling dialogues. This is the case in particular for implicit acceptance and rejection in the case of chained proposals and for the implicit acceptance in the case of phase change. These rules introduce new dialogue acts that have not been explicitly attributed to the previous utterances. For the rules of downward and upward inheritance, instead, no new dialogue acts can be inferred. It is rather the case that a dialogue act already available for the previous utterances is extended to new time objects. These rules therefore do not contribute to the discovery of indirect speech acts.

6 Related Research

Although there is a growing interest in the field of discourse processing and therefore in the area of contextual reasoning, not much work is reported about the use of contextual information in spoken dialogue systems.

Among the few approaches described the treatment of context in the p2 system (see [13]) bears most resemblance to our approach: the elements of which context is composed in p2 have been motivated from the specific requirements of a spoken dialogue system, like e.g. robustness and efficiency; from all the possible aspects of which context can be theoretically composed (for an overview see, e.g. [4]) only those aspects are used that are of immediate use by other system subcomponents. The context representation in p2, is composed of a task record that captures task-relevant information being exchanged during the dialogue, and a dialogue contents history which records the order in which the subtasks have been executed together with their propositional content. These two aspects of context can best be related to Thematic Structure and Sequential Memory in our system, respectively.

Relevant for our work is also the approach taken in [5]: this system processes dialogues with the input being analyzed thematically, intentionally, and interactionally. The results are stored in a short term memory, a working memory and a long term memory, the latter providing a frame-like representation of syntactic, semantic and pragmatic information.

Another approach that influenced the design of our Dialogue Memory is the three-tiered model proposed in, e.g. [10]: Linguistic Tier Information captures linguistic realizations of the concepts under discussion, Discourse Tier Information captures the speakers' model of the dialogue, i.e. of what has been said, and Belief Tier Information, finally, captures the cognitive state of the dialogue participants. While the information captured in the Discourse Tier can be related to the time objects of our Thematic Structure, the Belief Tier vaguely corresponds to the evaluation information attached to these objects.

Most of these approaches refer back to the three-level representation of discourse structure as proposed by Grosz and Sidner [6]; they distinguish (1) the intentional structure that describes the goals that are followed in a dialogue, (2) the attentional structure, that corresponds to the propositional content focused in the discourse, and (3) the linguistic structure which is related to the linguistic means used to convey this information.

7 Conclusion and Future Work

In this paper we presented a detailed account for the representation of context information in a spoken dialogue system. The Dialogue Memory which fulfills this task consists of a Sequence Memory to capture the chronological structure of the Dialogue and of a Thematic Structure that captures the task-relevant propositional information. We also showed, how the Thematic Structure supports inferences over time information and how these inferences can serve as the basis for the treatment of indirect speech acts in appointment scheduling dialogues.

Together with the dialogue component the Dialogue Memory is fully implemented and incorporated into the VERBMOBIL Research Prototype. As for the inferences induced by the processing of dialogue acts the system so far is able to capture
downward and upward inheritance. The implementation of the other principles proposed in this paper will follow in a later stage of the project. Additionally, we will examine our corpus for the occurrence of more phenomena that can be explained by means of inferences over the Dialogue Memory, and over the Thematic Structure in particular.

Since we plan to extend the VERBMOBIL scenario to a different domain we also will examine how the principles that have been identified here carry across to different applications. Depending on our findings we intend to develop techniques that allow an easy adaption of the inference mechanisms for other domains.

References


Dialogue Management in a Generic Dialogue System

Anders Baekgaard
Center for PersonKommunikation (CPK)
Aalborg University, Fredrik Bajers Vej 7, DK-9220 Aalborg, Denmark
e-mail ab@cpk.auc.dk

ABSTRACT

This paper presents an overview of the on-going work on a generic dialogue system that for a few years has been carried out at CPK, Aalborg University. The paper first describes the background for the generic dialogue system, and then describes its architecture, a dialogue description formalism and a dialogue manager. A methodology developed for description and management of mixed-initiative dialogues is briefly described. This methodology is reflected in a refinement of the existing dialogue description formalism (DDL).

1 INTRODUCTION

During the last few years work has been carried out at CPK, Aalborg University, on a generic dialogue system. This paper gives an overview of this work. Parts of the presentation in this paper are also found in [Dalsgaard and Baekgaard 94, Baekgaard 95a, Baekgaard 95b, Baekgaard 96].

The work on the system is aiming at establishing an environment for research in spoken dialogue systems, especially with focus on dialogue modeling, dialogue management, speech understanding and speech recognition, and with concern for multi-modal dialogue systems. Another aim of the work is - on a long term - to establish a broad generic dialogue system which eventually will provide tools for more efficient work on the otherwise time-consuming tasks associated with construction of dialogue systems, such as dialogue analysis, modeling and design, sub-language definition etc.

An important part of the system is the DDL (Dialogue Description Language) formalism for dialogue descriptions. The DDL is a language used by a dialogue designer to describe the various interactions between a user and the system. It is a compound language with visual (graphical), frame (slot-filling) and textual notations. The different notations reflect three levels of abstraction: The most abstract level where the overall structure of a dialogue is described, the medium abstract level where details of the dialogue are described by filling slots of frame structures, and the least abstract level where the details of the dialogue are described using a textual notation similar to a traditional programming language. The DDL language has abstractions for visual declarative description of dialogue structure, semantic rules, input and output syntax rules, and input and output lexica.

A dedicated graphical tool (DDL-Tool) which aids the dialogue designer in developing and maintaining dialogues is provided for the implementation of dialogues. The tool has a number of useful features such as syntax directed editing and source level debugging.

DDL dialogue descriptions are interpreted by an event driven generic dialogue manager (ICM - Interpretation and Control Module). It performs basic dialogue management functions such as interpretation of input (according to rules), output generation on the basis of rules, and automatic constraining of input devices on the basis of the current dialogue context. In addition, the ICM performs functions that are traditionally not associated with dialogue management, such as parsing of sequences of input events. The ICM simultaneously interacts with multiple input and output devices (such as speech recogniser, speech synthesiser, graphical displays) and multiple applications (such as databases) and an external NL parser module. During debugging of dialogues, it also interacts with the DDL-Tool.

A central part of the generic dialogue system is the GDS kernel (communication manager) which has overall control of the system. It manages system initialisation, the state of devices, allocation of devices to the ICM and system termination. It has a user interface to the dialogue designer.
which allows basic functions to be controlled. The GDS kernel is built using a set of C++ classes that are common for all modules in the system. The classes allow easy integration of event driven devices.

The work on the generic dialogue system focuses on problems seen from a technical point of view (as opposed to e.g. a cognitive or a linguistic point of view).

The generic dialogue system is originally developed within the SUNSTAR ESPRIT project and used for various speech controlled systems like a small CAD application, speech control of PABX functions, and services in the public telephone network. Later, the GDS has been and is being used in several applications in various domains:

- **Flight Ticket Reservation System**
  In the Danish project “Spoken Language Dialogue Systems” a demonstrator within the domain of flight ticket reservations was developed. [Baekgaard et al. 95, Bernsen et al. 95, Dalsgaard and Baekgaard 94]

- **Banking Services**
  In the ESPRIT project OVID [Larsen 96], a number of trial applications are developed within the domain of telephone based automatic banking services. The project focuses on simple but robust applications.

- **Book club**
  The book club [Larsen and Baekgaard 94] deals with porting of an existing voice-response-type of application into combined voice controlled and touch-tone controlled system.

- **REWARD**
  The REWARD project (http://www.kom.au.dk/CPK/Speech/Reward) is a relatively large language engineering project funded under the EU Telematics programme. The project aims at
  - Development of a platform for spoken dialogue systems (in the project called a “service creation tool”). The platform is a merge of the GDS and a commercial platform developed by Vocalis Ltd.
  - Development of a number of demonstrators in various domains: flight ticket reservation, public telephone network services, tele marketing and research (automatic interviewing systems over the telephone), and handling of incoming telephone calls to a PC repair service company (for routing the calls to the appropriate technical personnel).

The project will directly address the needs of the user organisations of the project (as opposed to the technology supplier organisations). It will equip them with an integrated tool to enable staff with a commercial orientation rather than a technical orientation to create new automatic telephone services based on spoken language dialogue technology and it will result in the creation and extensive field testing of a number of trial automatic telephone services in realistic environments.

- **MMUI**
  The MMUI initiative at Aalborg University [Mc Kevitt 96] addresses research and educational issues of intelligent multi media user interfaces. The GDS is being integrated with components from other local research groups in order to build a multi media, multi modal system capable of integrating speech understanding, image understanding, robotics, knowledge based medical diagnostics and dialogue management.

### 2 Dialogue System Architecture

The architecture of the system (figure 1) is designed to be modular and open such that it can be easily expanded, and all modules operating in the dialogue system adhere to a well-defined protocol. The protocol allows variable length packets of arbitrary contents to be exchanged between the modules.

The architecture allows the system to function in a multi-modal fashion which on the input side means that a speech recogniser can operate in parallel with a touch-tone telephone, a mouse or other input media, and on the output side means that synthetic speech or reproductive speech can be presented in parallel with screen text/graphics or other output media. The architecture is independent of applications and specific input/output devices.

During design and implementation of the architecture and the core modules, generality, flexibility and re-usability were major concerns.

The various components in the architecture are grouped into
Figure 1: Architecture of the dialogue system

- The dialogue component.

It consists of the Dialogue Description Language (DDL), the supporting DDL-Tool, and the ICM generic dialogue manager. The dialogue description formalism as manifested by the DDL is used by a dialogue designer to describe the dialogue between the human and the computers. It is defined as a formal visual language, and it is an recursive transition diagram-based language with compound visual and textual notations. Special features in the formalism are dedicated for spoken language dialogue. The formalism has several abstraction mechanisms for input and output which, at the highest level, allow the dialogue structure to be described in terms of semantic exchanges between the human and the computers. The ICM is the generic dialogue manager that manages the dialogue on the basis of the dialogue description.

- The input/output component.

It consists of devices that handle the various human-computer interaction modes, most notably the speech recognition device. In addition the application module is an architectural part of this group.

- The communication manager

A server module handles, besides the trivial routing of protocol messages, system initialisation and termination and controls the state of all attached devices. The architecture allows multiple simultaneous local or distributed instances of all modules, thereby allowing a system to be setup for multiple simultaneous users.

A part of the generic dialogue system is a set of generally applicable devices. A device for which a compliant GDS interface driver exists can connect to the system. Currently, the following devices are available: terminal (for text input/output), simple speech input/output (isolated words, reproductive speech output), Speech synthesiser (driver for an Infovox). In addition, other devices have been developed at CPK: SUNCAR (HMM based continuous speech recogniser, [Lindborg 95]), textrec (grammar constrained text recognition, [Broendsted 95]), TLI (telephone interface), graphical input/output (Open- Windows widgets, in progress).
3 The Dialogue Description Language

The dialogue description language is a language used by a dialogue developer to describe the human-computer dialogues with special regard on spoken dialogues. The dialogue description language is a compound language that consists of three levels: a graphical level (GL), a frame level (FL) and a textual level (TL).

The language that combines different description paradigms (visual notation, frame notation and textual notation) into a coherent language, and uses the various paradigms for what they are best suited for (see figure 2).

![Diagram of Dialogue Structure]

Figure 2: Abstraction levels in DDL dialogue descriptions

The GL was originally based on SDL (Specification and Description Language) which is standardised by CCITT [Belina 88]. It is basically a graphical language for describing state-event diagrams. It has been extended with new symbols, new meaning to existing symbols, new diagram types, and new ways of connecting symbols. The frame level and the textual level are not part of the standard SDL. The three levels reflect three levels of abstraction: the most abstract level where the overall structure of a dialogue is described using an extended subset of GL, the medium abstract level where details of the dialogue are described using the FL by filling slots of frame structures, and the least abstract level where the details of the dialogue are described using the TL.

The three levels may also be regarded as an explicit separation of the dialogue structure and the control structure, the declarative part and the computational part of a dialogue description, where the dialogue structure and the control structure is described using GL, the declarative part is described using FL or TL, and the computational part is described using the TL. The three levels are combined by annotating FL and TL descriptions to GL symbols, so for each GL symbol in a dialogue description, there is an FL and/or a TL annotation describing the specifics for that particular symbol in the given context.

DDL has the following major features

- Visual notation for state transition diagram-like dialogue structure description
- Abstractions for structuring dialogue description as separate sub dialogue diagrams
- Notation for definition of semantic representations (frames) for user utterances and for system messages
- Notation for declaration of categorial (non-feature) based input and output lexica
- Visual notation for declaration and use of rules for joining semantic representations and dialogue structure
- Visual notation for definition of context free input grammars, and output grammars
- Device and application independence allowing easy description of multi-modal dialogues.

3.1 The GL Symbols

The chosen set of the GL symbols are shown in figure 3.

The GL symbols is grouped into the following categories

- Modularisation and structuring symbols. These are start/stop/definition/end symbols for processes, procedures, input grammars, output grammars, rules and procedure (sub-dialogue) calls.
- State symbols. These denote waiting points for user input or other events.
• Input symbols.
These describe simple input events such as single words or touch buttons, grammatical input according to the a grammar definition, and semantic input that specifies allowable input according to rules.

• Output symbols.
These denote simple output events such as a single output message, and grammatical output that constructs an output sequence according to a grammar.

• Computation symbols.
These are task symbols for doing calculations, variable assignments etc., and decision symbols for branching the dialogue/control flow.

• Thread symbols.
These are symbols for describing task structure for multi threaded dialogue descriptions.

• Comment symbols - enabling attachments of comments to specific symbols are placed freely.

• Graphical layout symbols.
Waypoint symbol that is used to control the graphical "routing" of connection lines.

• Connection symbols.
These are arc symbols to connect symbols with ordinary flow, and dashed arc symbols to connect handlers.

3.2 INPUT LEXICA

The input lexica are divided into a number of sub-lexica that are referenced in sub-grammars and a global input lexicon that are referenced in sub-grammars or outside grammars.

Each input lexicon $L$ consists of an ordered list of lexicon entries $E_L$ which has a categorical reference point $P_L$ (denoted a pragmatic category) and consists of the tuple $E_L = [P_L, ev_L, D_L, I_L, H_L, V_L]$ where $ev_L$ is a functional predicate reference, $D_L$ is a device reference and $I_L$ is an item such as a text string or a variable, $H_L$ is an action program, and $V_L$ is a declaration of semantic slot-fillings as a list tuples $[cat_L, val_L]$ where $cat_L$ specify a slot in any SO (semantic object, see below), and $val_L$ specify the value to be assigned to that slot.

Lexicon access is through the $P_L$ and involves evaluation of the predicate $ev_L(D_L, I_L)$ on the basis of an object representation describing the
incoming event and possibly other context. A $P_{Li}$ can occur in more than one $E_{Li}$. The $H_{Li}$ program is executed only if the lexicon entry $E_{Li}$ is used for selecting a branch in the state transition algorithm (see later) and provide a mechanism for maintaining contexts.

### 3.3 Semantic Objects

A semantic object $SO$ represents a simple semantic value of input in terms of nested frame structures. The $SO$ is defined as a list of slots [name, type, default-value] where default-value applies if the slot is not otherwise assigned.

### 3.4 Input Symbols

Three different input symbols exist which reflects three abstraction levels of input. An input symbol is connected as a branch from a dialogue state, or it occur in input grammars.

- **Simple input symbols**
  A simple input symbol $IS$ refers to a lexicon entry through the reference point $P_{Li}$, and that can be an abstraction of simple events such as a click on a button, a DTMF tone, or if an isolated word recogniser is attached, a single word. Further, it can be system generated events such as a time-out or an error.

- **Grammatical input symbols**
  A grammatical input symbol $GIS$ refers to a context-free grammar. A grammar $G$ is defined using the GL, a consists of terminals which are simple input symbols $IS$'s, and non-terminals which are grammatical input symbols $GIS$'s. Recursive grammar definitions are thus possible.

An input grammar specify the type of $SO$ that represents the semantic value of the parsing results. Semantic values are assigned to slots of the $SO$ by matching the cat:$Li$ of the lexicon entries referenced by the $IS$'s of the grammar with slot names in the $SO$.

During runtime, the grammars are used by a recursive transition network (RTN) parser built into the dialogue manager.

- **Semantic input symbols**
  A semantic input symbol $SIS$ refers to a semantic rule $R$ that evaluates a semantic object $SO$. The branch of a semantic input symbol is taken if the rule it refers to evaluates to the largest scalar value in comparison with other rules.

A semantic rule $R$ is represented as a GL diagram that evaluates an $SO$ parameter and returns a scalar value (or nothing if the rule fails). The rule may access contextual data structures for the evaluation. A semantic rule declaration specify an $SO$ type and a list of NL sub-grammars that map semantic values on that $SO$. The NL sub-grammars are defined using an APSG formalism ([Povlsen 94]) or using DDL.

As mentioned above, the dialogue manager and the APSG parser are separate modules that interact through a simple interface. The dialogue manager gives the APSG parser the following information: a list of $SO$'s, a list of names of APSG sub-grammars, and the textual representation of an utterance as determined by a speech recogniser. The parser parses the utterance according to the set of sub-grammars and maps semantic values on the $SO$'s.

### 3.5 Dialogue States

In any dialogue state, branches to input symbols describe the input events that are expected to occur while in that state. Input symbols specify whether input events are treated as ordinary input or exceptions.

### 3.6 Handlers

Handlers are branches to input symbols from either the process start symbol (which is the start of the dialogue description) or from procedure start symbols (which are the start of sub-dialogues). A handler $H$ is attentive when the process start symbols or procedure start symbol from which it is a branch has been executed and the corresponding process stop or procedure stop symbols has not been executed (i.e. it is attentive while the sub-dialogue in which it is declared is active). An attentive handler can be invoked.

Handlers are either reverting or non-reverting. A reverting handler eventually transfers control to the dialogue state in which it was first invoked. A non-reverting handler transfers control to any position in the dialogue description in which case the dialogue manager restores its context to the context for that position.
3.7 Tool for the Dialogue Description Language

A support tool for the DDL (DDL-Tool) is used to create dialogue descriptions. The tool has an easy-to-use user interface which allows dialogue design to be performed off-line in a top-down manner where the dialogue structure is first defined on the GL level, and the declarative details are added to the FL using a set of frame structures to be filled in, and the computational details are added at the TL. The tool has the following major features:

- Direct graphical manipulation of GL with automatic (default) handling of layout of connections
- Object references (sub dialogues, lexicon entries, grammars etc.) are used as hyper links to their definitions
- Instant syntax check, verification of compliance to DDL definition
- Online graphical debugging
- Printing of diagrams in DVI format

The online graphical debugging implies that the tool can be used as a source level debugger. It traces the execution of a dialogue and highlights the individual GL symbols as they are executed. In addition, the tools allow the designer to inspect data structures and to control the execution of the dialogue by setting and clearing breakpoints, single stepping, stepping over subdialogues and jumping to a specific symbol.

4 THE ICM Dialogue Manager

The ICM dialogue manager is the central component in the GDS architecture. It interfaces via the communication manager to the input/output devices and the application. In addition, it interfaces to the DDL-Tool in order to allow source level debugging. The DDL dialogue description is compiled by the DDL-Tool into a textual language which is subsequently read by the ICM.

4.1 Functions of the Dialogue Manager

The ICM dialogue manager performs the following major functions

- Interpretation of DDL dialogue descriptions
- Parsing of input according to grammars defined in DDL (it can also interface to external NL parser modules)
- Logging of user interactions for assessment purposes
- Control of input and output devices and the application
- Debugging functions in conjunction with the DDL-Tool.

4.2 State Transition Algorithm

The state transition algorithm specifies what actions are taken by the dialogue manager while waiting in a dialogue state and an input event is arriving. Before the dialogue manager enters the waiting state it computes the constraints (see later) and imposes these on the input devices. In addition, the relevant parts of the constraints together with the set of SO’s are handed to the APSG parser (which then prepares itself for parsing). When input arrives, the dialogue manager performs the following actions.

1. If RTN parsing status is set, continue parsing with the input event as the next input token. If parsing completes, select the branch corresponding the the input grammar that was used to complete the parse.

2. If input is classified as a continuous speech utterance invoke the APSG parser and retrieve the SO’s.

3. If any SO’s have filled slots, then for each of these SO’s evaluate the semantic rule that knows about that SO.

4. If any rule evaluates to a value, then select the branch corresponding the rule that evaluates to numerically highest scalar value and terminate the state transition algorithm.

5. Invoke the RTN parser with the input event as the first token. If parsing is possible according to any input grammar, then set the RTN parsing status and terminate the state transition algorithm.

6. Evaluate each of the IS, and select the branch of the first one that evaluates to true and terminate the state transition algorithm. The order of evaluation is such that for all $E_{Lx}$
referenced by the IS's, $E_{L_4}$ is evaluated before $E_{L_j}$ if $i < j$.

7. For each attentive handler, considered in reverse order of becoming attentive, evaluate the SIS, GIS, and IS for that handler as specified in steps 3 to 6.

When the state transition algorithm terminates, one of the following actions are taken

- If any branch was selected, transfer control to that branch.
- If RTN parsing status is set, calculate new constraints and impose them on the input devices.

4.3 Constraining of Input Devices

The dialogue manager performs constraining of input devices in order to increase performance in both terms of accuracy and processing time requirements. The effects of constraining input devices are

- It improves the accuracy of the speech recogniser since the search space is smaller.
- It increases the potential maximum size of the language model because only parts of the language model need to be activated in a given dialogue state. For example, in the flight ticket reservation system there is approx. 600 words forms, but at most 110 words forms are active in any dialogue state.
- It allows visual feed back to the user of possible input for graphical input devices as user actions that are not possible can be inactivated ("grayed out").
- It increases the processing performance (speed) of the speech recogniser.
- The constraints can be applied also on the NL parser to increase processing performance.
- It insures correspondence between possible input and the dialogue description thereby eliminating the need for error handling in the dialogue description.

Constraining of the speech recogniser increases the risk that a user utterance is misrecognised.

The constraints are calculated on the basis of the dialogue model and the current dialogue context and constitute the set of input events that can be handled by the dialogue manager.

4.4 Managing Mixed Initiative Dialogues

All systems described in the introduction were strongly system directed. A methodology is currently being developed that allows for description of mixed initiative dialogues.

A dialogue is described as a set of independent tasks and a task structure, where each task is a DDL diagram and the task structure is described as a DDL diagram using dedicated primitives (DDL symbols). Each task description can access and update data structures representing the context. The task structure diagram specifies the relationship among the tasks in terms of which tasks can be active in parallel and which tasks can be executed sequentially. Figure 4 shows an example.

![Task structure diagram](image)

Figure 4: Task structure diagram

The tasks $T_1$, $T_2$ and $T_3$ are active in parallel. When task $T_3$ is terminated, task $T_4$ becomes active. When both $T_1$ and $T_2$ are terminated, $T_5$ becomes active. When both $T_1$ and $T_2$ are terminated, all tasks are completed. The tasks are "joined" in the $J_1$ and $J_2$ collapse symbols. As an example, $T_1$ is the task that inquires a "departure time", see figure 5.

The flow of control in the task is as follows: execution starts in the Departure time symbols and then enters the await input state symbol. User input that complies to either the time rule or the departure time rule causes control to be transferred to the update context task symbol followed by
termination of the task. The got-focus and time-out meta events are generated automatically by the dialogue manager. The ask-departure output symbol generates a prompt for the user.

The time semantic input symbol refers to a time rule that describe the semantic structure and (indirectly) the syntax for a time utterance. The rule for departure-time is specific for a departure time utterance (e.g., “I want to leave at 8.30”, whereas the time rule allows e.g. “at 8.30”).

The dialogue manager evaluates all rules referenced in the set of active tasks and selects the rule that produces the best score. For rules that produces equal scores, the one belonging to the task currently in focus takes precedence. In the example above, the departure time rule gives better score than the time rule and can cause focus shift from another task (possible also using the time rule) to the departure time task only if user input is sufficiently specific.

The dialogue manager treats each task as a thread in a multi-threaded environment.

5 CONCLUSION

The present paper describes the architecture and the intrinsics of a generic dialogue system (GDS). The system has evolved during more than a 5 year period and during this period, the platform has been applied to several systems of various nature. In the ”Spoken Language Dialogue Systems” project, a complex system in the domain of flight ticket reservation and information has been built and evaluated. A project within CPK has dealt with porting an existing "voice response" class of application (Book club services) to a combined voice controlled and telephone touch tone controlled application. The results of evaluating these systems and their development show that the GDS is suitable for simple as well as more complex systems. Most problems encountered were with respect to robustness of the speech recognition, and that the human computer dialogue is conceived rigid due to the strong system direction.

The existing systems were based on spoken language as the only interaction mode (an exception is the Book club system which in addition accepted telephone touch tone input). Future applications in office or laboratory environments (complementary to applications in telecommunications environment), as planned for the MMUI project, will draw on a range of interaction modes and input/output media such that spoken language is one modality in combination with i.a. mouse, tactile sensors, video cameras and graphical displays based interaction modes. Currently, a project within CPK deals with an application where graphical media are used for the interaction, and adjustments of the GDS are implemented.

The systems built so far are restricted in terms of functionality and are in limited domains. As related technologies (such as speech recognition and visual image interpretation) improve, the possibility of integrating the various components into large complex systems opens. This requires scalability for the underlying architecture and methods. The current GDS architecture allow for multiple instances of components of various nature. The question is how to describe the inter-dependency and temporal bindings between the individual modalities if each mode of input span a large event space.

REFERENCES


DIALOG MANAGEMENT IN THE CMU
SPOKEN LANGUAGE SYSTEMS TOOLKIT

Wayne Ward
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA

ABSTRACT

Carnegie Mellon University is assembling software that we have developed over the past several years into a Spoken Language Systems Toolkit. The intent is to provide the tools required for a developer to build a conversational style spoken language interface to limited domain computer applications. The system is specifically designed to support information extraction from spontaneous speech. We support a simple dialog structure aimed at trying to acquire the information required to perform the actions desired by the user. The system supports mixed-initiative interactions. It must engage in clarification dialogs when input is corrupted, ambiguous or inconsistent. It should prompt for information if the input is incomplete or if the user is having trouble. It should also help the user understand the capabilities of the system and stay within its constraints. One of the most serious sources of error in Spoken Language Systems is out-of-domain queries by users. The goal is to provide as simple a system as possible for developers that still has enough capability to provide a conversational interface for users to limited domain applications.

INTRODUCTION

Spoken Language researchers at Carnegie Mellon University have been active in building spoken language systems for research purposes for many years. We are in the process of assembling the research speech understanding modules we have developed into a toolkit. The toolkit is for use by developers who would like to add speech interfaces to their applications. It will be especially useful to those developers who would like to provide conversational voice interfaces.

The features included in the toolkit are those that we have actually found useful in the process of creating the applications we studied. It has grown in a bottom-up fashion with features being added as they are required, rather than being guided by a grand design of what features we thought a priori should be included in a system. After deciding which features we wanted to include, we did modify the system design so that the implementation could be consistent and efficient.

We have built spoken language interfaces to several applications. The most complicated of these was the Air Travel Information Service, the ARPA common task for evaluating spoken language systems. In this application, users get information from an air travel database by voice. In implementing the ATIS system, we were surprised at where most of the development effort was required. We had expected that most of the effort would go into developing a semantic grammar with good coverage of the spontaneous utterances of naive users. While grammar coverage was the largest source of error, it did not require the most effort. The grammar rules are quick and easy to add. They don't have much of a ripple effect, that is, adding new rules doesn't tend to break old ones. The maintenance of context or constraint propagation required the most time and effort to develop. It did tend to suffer from a ripple effect and required careful thought. This led us to extend the system to provide a framework and some functions to support dialog and context management.

CONVERSATIONAL MODE

Our notion of a conversational interface to an application is one in which the user interacts with the system as if it were a human assistant who was an expert on the application. This includes asking for help and engaging in clarification dialogs to correct misunderstandings. It also means the system must be able to guide users when they are unsure what they want or how to use the system. The success of a voice interface often depends on how well the interface is able to keep the user inside the domain. Out-of-Domain utterances are a major source of error
in such systems.

The application must know enough about the domain to understand what the user is saying and how it relates to the domain. It is a goal of our system to provide simple mechanisms for a developer to specify the necessary domain knowledge to implement this type of interface.

**SYSTEM ARCHITECTURE**

The general architecture of a conversational application interface built using the toolkit is shown in Figure 1. The speech recognition component produces a string of words (it can also provide an N-best list if desired). This string is first passed through a string mapper which is used to map problematic expressions to a desired form. An example is mapping PHILLY to PHILADELPHIA which is the string which must be matched against the database.

![Diagram of Interface Architecture](image)

**Figure 1**

Architecture of Interface

The string is then passed to the parser which produces a rank ordered set of semantic frames. A frame contains a semantic representation of the important information and is referred to as a gist. If the best interpretation is ambiguous, the dialog module is used to disambiguate and select a single interpretation. The confidence in the interpretation is then assessed. If confidence is too low, a clarification dialog is initiated. Otherwise, the frame is then put into canonical form by mapping the values of selected semantic tokens, like putting dates and times in a standard form. This is necessary because the way in which users specify information may not match the way it is required by the application. Frames are constructed to model what users say and canonical frames model what the application needs (in the way of information). These aren't different types of data structures, but a set of functions to input a semantic tree and output a modified version of it.

The canonical frame is then merged with prior context to form the final interpretation. This final interpretation represents the current context of the system. This context is examined to determine if sufficient information exists to take the specified action. If so, the action is taken. If not, the user is prompted for missing information.

**TOOLKIT ELEMENTS**

The CMU toolkit is designed for the development of simple, robust gisting and conversational interfaces to applications. It is oriented toward gisting limited-domain conversational speech. It contains routines for Speech Recognition, Parsing and Dialog Management. These usually are configured as two process, a speech server and an application, interacting over ethernet. The parser and dialog manager are included in the application process.

The Sphinx-II [1] and Sphinx-CLM systems are the speech decoders used. These differ in the type of language model that they use. The speech decoder must be given a lexicon and language model for the application. The lexicon is the set of words the system knows and their phonetic pronunciation. Language models are used by speech recognition systems to assign a priori probabilities to strings of words. These are traditionally word based trigrams developed from a corpus of training data. This is the case for the Sphinx-II system. The Sphinx-CLM system uses a Class Language Model. A class language model is a stochastic language model in which transitions to tokens, referred to as classes, are modeled. Generally, classes expand to a list of single words, hence words are thought of as being in a class. This is too restrictive in many situations, transitions are still segmented and modeled at the word level. We have extended the notion of a class to be expanded by a Recursive Transition Network. This allows not only single words but strings of words to map to a token. Thus, a string of words that represents a concept in the domain can be modeled as a unit. The language model has a stochastic representation of the sequence of classes and a grammatical representation for expanding the class to words. The RTNs used to expand the classes are a subset of those used by the parser. The toolkit contains routines to train both types of language models.

The toolkit uses a robust parser derived from
the Phoenix [2] system. The parsing is done by a C function. This function is passed a pointer to the string to be parsed and it returns a pointer to a frame structure. This structure can be printed to a string as a bracketed string or passed directly to the application. The operation of the parser is to produce a gist, or extract the relevant information from the input. Frames are used to represent semantic elements. A frame represents some basic type of speech act or action for the application.

![Diagram](image1.png)

**Figure 2**
Example Parser Output

Slots in a frame represent the information that is relevant to the action. Slots are filled by matching patterns in the input word string. Semantic grammars are used to specify how word strings map onto slots. Each semantic token has a grammar which specifies the patterns (strings of words) that are an instance of the token. Slots are root-level grammars that can match a string independently of the context around it. The parser matches slot-nets against sub-strings of words in the input. When a slot-net matches a substring, it is passed along for incorporation into frames. The slots are filled independently of the order in which they appear in the frame. Each instantiated slot is the root of a semantic parse tree. An example of the parser output is shown in Figure 2.

How loose or tight the system is, is determined by how the grammar is written. If it uses one frame with one slot, it will be a standard RTN parser. This is efficient, but not very robust to unexpected input. At the other extreme, making each content word a separate slot will basically give a key-word parser. Somewhere in between usually gives the best combination of robustness and disambiguation. We usually write slot grammars at the 'semantic phrase' level.

The toolkit contains routines to:

- Generate grammar files from a corpus of annotated examples. These files may also be created directly with an editor.
- Compile grammar files into Recursive Transition Networks.
- Parse text input using the grammar specified as a command line parameter.
- Tag input for building a semantic class language model.

It also contains a library of grammars for general concepts like time, date, number, etc. These tokens can be used by developers and then loaded at run time.

The system has hooks for the functions to map structures to canonical form. The developer specifies a list of semantic tokens and provides a C function for each. From this the system builds a table associating specified tokens with pointers to functions. These functions input a pointer to a parse subtree and replace it with an altered version. An example of this process is shown in Figure 3. There are generally only a few tokens that require this type of mapping.

![Diagram](image2.png)

**Figure 3**
Example of Canonical Mapping

**DIALOG MANAGEMENT**
A wide variety of functions are carried out by the modules referred to as the Dialog Manager:

- Disambiguation
- Constraint Propagation
- Clarification / Mixed-Initiative
- Repair
- Inference
- Generation of Expectations

These are the operations we have found most useful for the applications which we have built.

Disambiguation covers the functions having to do with establishing a preference order on a set of parses based on information other than that contained in the parse itself. This includes anaphora and ellipsis which are resolved via a history list. The history list is the list of the N most recent canonical frames in the session. The parameter N is configurable at run-time. It is often the case with these limited domain interfaces that resolution of anaphora and ellipsis is trivial as there is only one reasonable meaning or referent. When the referent is ambiguous, the history list is consulted. An anaphoric referent is the most recent thing talked about that could possibly serve. In the limited domains we have implemented this resolution is almost always trivial and the anaphoric token can be ignored without changing the interpretation.

This system has a unique way of treating ellipsis, which is, it's nothing special. The system has no notion of sentence structure. Input is normally treated as a sequence of semantic fragments. These are in some sense all elliptical expressions. Each is merged with the others into a common interpretation. In addition to a history list of frames the system maintains the current state in the form of a set of canonical frames. This represents current active information and constraints which are the result of merges, repairs, overwrites, etc. Each new frame coming in is merged into the current state. Elliptical expressions fill slots in frames which are then merged into the current context the same as any other frame coming in. When merging into the context, if a value already exists for the token being added, the old value is overwritten.

Maintenance of the current context involves constraint propagation and requires some amount of inference and/or heuristic rules. As an example, suppose the user says "Flights on Friday morning before eight" and then after looking at the output says "Flights on Friday morning". The assumption is that the second utterance cancels the "before eight" constraint (but maintains all others such as the route). What if the user says "First class flight Friday morning before eight" and then "The cheapest flight". Does this mean "the cheapest first class fare Friday morning before eight" or simply the cheapest fare between the two cities, or some set of constraints in between? Some such questions can be settled in a general way by general rules applied in conjunction with a simple knowledge base. For example, after "flights at 3 pm", the phrase "afternoon flights" means at any time in the afternoon, replacing the three pm constraint. Even though these represent different low level tokens [at_time] and [time_period], they both map to the same field in the canonical frame, so it is trivial to replace the old with the new. But it is also possible to determine that one is a more specific version of the other, and to use a general rule of the form "a more specific constraint following a more general one adds the new constraint" and "a more general constraint after a specific one clears the specific constraint". The system provides a simple knowledge base framework to allow simple inference, but it also allows the developer to specify a specific set of tests and associated actions to manipulate the current context. An example might be "if a new depart location and arrive location are mentioned, clear all previous constraints".

Clarification involves several different types of interaction with the user. Acoustic Verification, Language Ambiguity, and Underspecification. Mixed-Initiative interaction is an extension of these same functions in which the system assumes the initiative and prompts the user for the information needed.

Acoustic Verification is unique to spoken language interfaces. The speech decoder and parser each provide confidence estimates as to how reliable the interpretation is. These estimates are combined to give an estimate of the reliability of the interpretation. Portions of the interpretation (including the whole thing) may be suspect or incomplete. If the confidence in the interpretation is below a certain threshold, the user is asked to verify the interpretation. This may be a single piece of information or the interpretation as a whole. If it is not correct, then they engage in repair behavior. If the confidence measure is below a second threshold, the interpretation is rejected. Again, this may be a single piece of information or the entire utterance. A focused dialog is then carried on with the user to try to obtain the correct information.

Language Ambiguity refers to the common situation where there is more than one reasonable interpretation and the system has no way to prefer one over the other. The user must be presented with the alternatives and asked to pick the correct one (or
Underspecification refers to the situation where the user has not given the system enough information to complete the specified action. It may be that there is enough information to take the action, but that the output volume would be too great, such as an underspecified database query. The query "Show flights from Boston to California" would probably generate more output than the user wants. This factor is even more important over more limited bandwidth channels like the telephone. We have adopted a simple mechanism of allowing fields in the frames to be marked as required. Prompts are associated with each field. When a required field is missing, the system prompts for it. The user can override the requirement. It is not always clear ahead of time that a query will be underspecified. Some similar queries in a database query system could return very different amounts of output. In our database query systems, we build the query and check the number of records returned before presenting the data to the user. If the number of records is greater than a preset number, the user is told the number of records and prompted for specific information to narrow the query. The user is not required to give the information promptly for, and can give other information or demand all of the output to the current query.

Mixed-Initiative is used to mean the style of interaction where the user is allowed to take the initiative and speak to the system in a free form. If the system detects that the user is having problems, it takes the initiative and starts prompting the user for information, generally following a flowchart or menu system.

Repair refers to interactions in which the user makes a correction. The cause of the error could be either the system or the user. It is treated in a manner similar to ellipsis. This generally consists of replacing one or more tokens extracted from the current or previous utterance with a new value. Things like "I want to go from Houston, no Dallas to Denver" and "I said Dallas not Dulles" or "Change Houston to Dallas". It is usually easy enough to determine what the new piece of information is. The main difficulty is in determining that the new information is to replace some current piece rather than be added to the current set. This is normally done in our system by having the semantic grammar detect the cue phrases which signal the repair. Things like "I meant X not Y", "sorry", "no", "oops", even filled pauses like "uh".

Inference for us was just making decisions based on the relationships between concepts. This is accomplished through a simple knowledge base having ISA, PART_OF and PROPERTY links. This mechanism is used is ellipsis resolution, constraint propagation and reasoning about the meaning of a user request in terms of how it maps onto the system's representation. Even users who are familiar with a domain do not necessarily know how the system represents it. For example, in the ATIS domain, it is common for users to ask for first-class flights. In the database, first-class is not a property of flights, but of fares. Flights have fares, and some fares are first-class. So the system has to realize that first-class flight means a flight with first-class fares. While this is only very simple reasoning, either there must be a general mechanism for it, or the developer must explicitly provide the appropriate mapping to the system representation.

**Use of Expectations**

We would like to use the information available to the dialog manager to improve the performance of the speech decoder and parser. Since these modules are basically stochastic search engines, it is convenient to provide information to them in the form of a probability distribution across semantic tokens, given the current state of the system. The Sphinx-CLM decoder is well suited to this style of interaction. It's stochastic language model consists of a subset of the tokens used by the understanding portion of the system. When it comes to the end of the pattern representing a token, it accesses its language model to get the probability distribution for the next token, given the history. By default it will use a token based trigram to estimate the probability distribution of the next concept given the history. However, these probabilities may be provided by any mechanism; ngrams, stochastic networks, goal trees, etc. The requirement is that some method be used to generate expectations of the next concept in the form of a probability distribution. These probabilities are then combined with the acoustic scores in the decoder to drive the recognition of word strings. They are also used by the parser to put a preference ordering on possible parses. Now that we have the mechanism to support the use of such probabilities, we are incorporating concept expectation algorithms from the MINDS [3] and MINDS-II [4] systems and training them to produce probabilities.
ACKNOWLEDGEMENTS

The work described in this paper was done in collaboration with Sheryl Young. Much of the design was motivated by a desire to incorporate the principles of her MINDS and MINDS-II systems as an integral part of the toolkit to facilitate experimentation with spoken dialog. Sheryl died shortly before preparation of this manuscript began.

References


UNDERSTANDING OF SPONTANEOUS UTTERANCES IN HUMAN–MACHINE–DIALOG

Wieland Eckert

Lehrstuhl für Mustererkennung (Informatik 5), Universität Erlangen-Nürnberg, Martensstraße 3, D-91058 Erlangen, Germany
email: wieland.eckert@informatik.uni-erlangen.de

ABSTRACT

In this paper discuss three basic problems of current spoken dialog systems: (1) the problem of understanding speech, (2) the additional problems imposed by spontaneous speech, and (3) the problem of dialog processing. We describe different definitions of the term understanding and propose some theses for an interpretation system. We show two principal methods for enhancing the robustness against phenomena of spontaneous speech. Then we discuss several definitions of the term dialog used within the speech community. Interpretation of utterances within a dialog context is necessary to resolve ambiguities. After that we discuss some factors of dialog control that have great influence on the next user utterance. Since evaluation of dialog systems is not yet standardized, we show the definition of a measure for the systems capabilities to understand utterances. This measure can be calculated automatically. Finally, we report some figures for our own spoken dialog system. The references given in this paper are expected to be a quite comprehensive starting point for further readings.

1 INTRODUCTION

Currently, we see a fairly large number of automated systems upcoming which pretend to guide a natural spoken dialog with a human [28, 32, 5, 30, 2, 3, 7, 9, 14, 17, 19, 24]. Unfortunately, everyone has got his/her own meaning for important things like interpretation, understanding, dialog or quality measures. These different views of the world are mainly caused by the fact that there is no commonly agreed standard for applications nor a common benchmark for dialog systems.

In this paper we want to shed some light on the terms understanding and dialog. While some of the theses given in the paper might be provocative, we aim to start discussions in the community about proper (and commonly agreed) definitions. In the following section we discuss possible definitions and properties of the term understanding in the context of spoken utterances. Aspects of understanding spontaneous speech are treated in section 3. In section 4 we discuss the term dialog in the context of spoken human–machine dialogs. After that we summarize in section 5 the current approaches to rate various aspects of dialog systems. In section 6 we present a short description of our own demonstration system and present some evaluation results.

2 UNDERSTANDING SPEECH

A typical spoken dialog system consists of a word recognizer, a parser, and a dialog manager. While there exist different approaches (like an integrated knowledge base coupled with a search mechanism, compiled network, blackboard architecture), most of the systems mentioned above utilize this kind of modular approach. Considering these modules we can ask the questions: what is understanding and in which module is it performed? Experts in different fields have different views of the understanding process. Some of them are:

Understanding = recognizing the word sequence. An utterance might be called understood when the correct (i.e. actually spoken) sequence of words was recognized. From this point of view the words are to be identified with their meaning.

Understanding = building internal structure. Here a sequence of words is seen as just
interpret semantic information. We need to separate data and algorithms.

We assume that a parser returns (mainly/only) semantic information about the utterance. Thus the information exchanged between the parser and the dialog manager has to be represented in some semantic language. On the other hand the interpretation process that "makes sense" of the users utterance has to consider dialog context and world knowledge. Therefore an interpretation process utilizing different knowledge sources is started on the semantic representation — resulting in the understanding of the utterance. Coupling the representation formalism and the interpretation mechanism in a compound knowledge base might cause difficulties regarding the maintenance of the system.

These 3 For semantic description of utterances we need an adequate level of representation — not too simple and not too complicated.

We hope that nearly every knowledge engineer would agree that finding a proper representation formalism is not a science but an art. In principle all representation formalisms are supposed to be of equal power. But there is never the right one.

These 4 It is not useful to represent or interpret all possible relations of objects. Only a small part of them is meaningful and relevant in the dialog context.

While there are approaches to make up the most general knowledge base of the world, we think that spoken dialog system does not really need to deduce everything. Applying large amounts of world knowledge would lead to increasing sets of ambiguities which are meaningless within the current domain of the system. Restriction of the system capabilities to a certain (small) world increases the effectiveness for "proper" dialogs that do not leave the application domain.

These 5 Idioms and phrases need to be described as a whole. Ambiguities that are generated by taking the verbal interpretation are (usually) unintended.

Idiomatic and phrasal expressions are used that often in (spontaneous) speech that they deserve simplified processing and could be easily excluded from ordinary linguistic analysis. A simple pattern matcher can assign a semantic interpretation to phrases like May I ask you a question? without performing expensive analysis steps.

These 6 Utterances containing the same meaning should have the same representation, utter-
ances containing a similar meaning should have a similar representation.

Apart from the idiomatic and phrasal expressions, the composition principle is a basic property of the language: further descriptions of objects are simply performed by attaching PPs or by relative clauses. Therefore the resulting semantic representation should reflect the minimal change imposed by this description by having only small parts modified. Thus, the composition principle should be employed into the semantic representation formalism.

These 7 Understanding utterances can be performed by simple deduction rules with local scope, together with the generation of references into some environment.

An environment is a suitable place to store initial world knowledge as well as dynamic referents. Given a structured semantic representation induced by the composition principle, we claim the existence of simple deduction rules. The interpretation process is applying these rules and results in chains of deduction steps.

These 8 For the representation of elementary actions a small number of primary types is sufficient. Further distinction is performed by additional attributes.

In a particular application domain we just need to represent a few relevant types of actions. According to [27] it is appropriate to have only 12 of them. While this might be a quite domain dependent design decision, we still believe that a small number of primary actions is sufficient.

These 9 Not every ambiguity has to be resolved. Ambiguity in utterances might be present but irrelevant. Ambiguity might be used intentionally or systematically and must be preserved in that case.

A well known example for a structural ambiguity is the sentence I saw the man with the telescope. While it is hard to tell the owner of the telescope, this sentence can be translated easily into, for instance, German — retaining this ambiguity. However it is a quite hard problem to decide automatically, whether some ambiguity has to be resolved or might/must remain present in the resulting semantic description.

These 10 An understanding system is nonmonotonic. There is no "proven" knowledge; "facts" make only sense with respect to their context.

Utterances or even parts of a single utterance might be contradictory. Since a speaker could not be forced to talk in first order logic, we have to expect contradictions. Self repairs within spontaneous speech (cf. next section) are a special case of contradiction. In the interpretation mechanism there must be provisions to revise or even “forget” objects or attributes.

These 11 An interpretation system is incomplete. There are always propositions which could not be interpreted.

Since individuals have different models of the world, there is currently no chance to find the most general model. Considering the current state of the art it is useful to limit the system's capabilities to a certain small domain and a simple task. We need to accept that a spoken dialog system is allowed to fail.

These theses have quite some impact on the resulting system. By considering them we get a clearer idea of the capabilities and limits of the overall system as well as its components. Obviously, some of these theses could be discussed. This is what they are made for!

3 Spontaneous Speech

Spontaneous speech differs from clean language and in the theses shown above there was no provision to deal with specific phenomena observed in spontaneous speech. Common effects are (cf. [25, 33]): elliptic utterances, irregular word order, self corrections, restarts, or utterances containing multiple sentences. These effects are more often observed than regular, grammatical utterances. Everyday speech does not follow the hard rules of grammar. A more detailed analysis shows that:

- prosody and speaking speed differ from read speech,
- utterances follow a quite simple pattern with low linguistic complexity,
- users' creativity in building new utterances is very limited, they use the same words as the system (parrot syndrome) or they complete system utterances using ellipses, and
- effects of false starts, hesitations and self corrections are not systematic — they can happen at every word position within an utterance.

footnote: This model must include the idea of self reference — another difficult problem.
Considering these findings, we need special provisions to automatically understand spontaneous utterances. First of all, the word recognizer has to be trained with real data, i.e. data containing an appropriate amount of these irregular utterances. Both steps of training the word models as well as the language models benefit from a sample of spontaneous data. Variations in prosody and speed are mainly incorporated into the word models. The other effects mainly influence the resulting language models. Recent advances in the field of speech recognition show that the language models have a large impact on the recognizers accuracy.

A major problem is the linguistic analysis of the effects of spontaneous speech described above. We assume that the linguistic analysis is performed by a parser, which utilizes a lexicon and a grammar. For the analysis of ungrammatical input there are two different directions:

- All variations of expected ungrammaticality are analyzed and a grammar of spontaneous speech is build by merging these additional rules with a grammar of written language.

- The grammar only contains proper rules and all ungrammaticality has to be handled by the parsers ability to deal with partial parses.

The first case seems computational expensive since it allows nearly arbitrary combinations. Using a search mechanism we have to find the best parse out of many different “ungrammatical” (but modeled!) continuations. Considering the possibility of misrecognition within the acoustic recognizer, this approach would always find an interpretation — even when processing garbage input. Actually, this approach is counterproductive when we consider the word recognizer to deliver not only the best word string but a word lattice or a word graph. We can easily imagine that due to a few rules of spontaneous phenomena the search space explodes.

A robust linguistic processor needs to be able to analyze partial utterances and to represent partial parses. In this case the grammar contains the clean theory not extended with rules to cover spontaneous phenomena. It is up to the parser to find maximal consistent subsequences in the recognized word string or word graph. For that purpose the parser utilizes a lexicon and a set of grammar rules that define possible combinations of words as well as the semantic representation of the phrases resulting from these combinations. Traditionally, a parser can either analyze a given input with respect to the underlying grammar or it fails. Thus, all spontaneous speech phenomena that are to be understood by the system have to be modeled in the grammar. Apart from the fact that it is quite unrealistic to foresee all types of errors, corrections etc. this approach becomes prohibitive when the word recognizer delivers a word graph instead of the best word string. Using such an interface it is the task of the parser to find the best scoring grammatical(!) path through the graph. But if the grammar models ungrammatical strings that may occur in spontaneous speech, the grammar becomes worthless for separating grammatical from ungrammatical paths through the graph. Furthermore, this approach is computationally too expensive since it allows nearly arbitrary combinations.

Therefore a less rigid parser must be used which allows partial parsing if the grammar does not permit a complete analysis of the input. Such a robust parser does not fail if no result spanning the whole input can be generated but delivers one or more partial results instead. These partial results represent grammatically well-formed utterance fields. A sequence of such utterance field objects (UFO) can then be handed over to the dialogue manager which tries to combine the parts using contextual knowledge.

4 Human–Machine–Dialog

In this section we want to clarify the third of the keywords given in the title of this paper. Please keep in mind that we restrict ourselves to task oriented spoken dialogs between a human and a machine. For the moment human–human dialogs or multimodal dialogs are not considered.

4.1 What is a Dialog?

In a dictionary [1] you get two basic definitions of the word dialog: a conversation between two or more people and an exchange of ideas or opinions. Since the term dialog is explained by other terms (conversation or exchange) we can not make use of this definition for our problem.

On the other hand, several so called dialog systems have already been built. They are not comparable since every system developer has a different view of a dialog regarding quality of a dialog or power of their systems. In the following we distinguish three classes of dialog systems. Their

-----

2 The other definitions are not relevant in our context.
Table 1: Different types of dialog systems.

<table>
<thead>
<tr>
<th>Type</th>
<th>Initiative</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>system</td>
<td>direct answer, no history</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>user</td>
<td>direct answer, limited history</td>
</tr>
<tr>
<td>Conv.</td>
<td>mixed</td>
<td>answers, questions of both partners, large history</td>
</tr>
</tbody>
</table>

The main difference lies in the role of the dialog initiative and the sort of expected response. An overview is given in Table 1.

**Menu Systems** are controlled by the system. All system utterances are of the kind Do you want choice a, b, or c? and the user is just allowed to answer directly. A typical example for this kind of systems is the automated hotline telephone service of a large company, which is typically based on touch tone recognition in spite of the presence of speech recognition technology.

**Question & Answer Systems** are designed in a way that the user takes the initiative and formulates a complete request. The response is a set of data that fulfills the request. In rare cases it is possible to refer to the result of the previous request. Typically, a dialog consists of one or two user turns. The original ATIS systems meet this definition.

**Conversational Systems** have the ability to move the initiative from one dialog partner to the other one (and back again, of course). Dialogs contain several turns and they contain requests, answers, clarifications, confirmations, and so on. In conversational systems the reaching of a complex goal is split into several steps which are performed in sequence, whereas in Q&A systems these steps are combined into a single exchange. The later conversational ATIS and the family of the SUNDIAL systems are typical examples for this class.

The given order shows increasing dialog complexity. The systems tasks are extended, too: the interpretation process needs to apply more knowledge for understanding an utterance, and the conversational capabilities of the dialog manager require more elaborated dialog models and access to the dialog history. Obviously, the class of conversational systems is the desired solution for speech communication systems. For other modalities we already have some examples of successful systems (e.g. automatic cash machines, flight schedules) in our everyday life. In the following we concentrate on conversational dialog systems.

With this view of a dialog system we get back to the four different definitions of the term “understanding” shown in section 2. The only plausible definition remaining is that of a proper system reaction. Figure 1 illustrates this point of view. In the following two subsections we discuss the remaining problems of interpreting an utterance and finding a proper system reaction.

### 4.2 Interpreting Utterances

Interpretation of an utterance can be separated into several steps:

- A sequence of semantic descriptions is constructed by a parser which applies linguistic knowledge to the recognized word string or word graph. As shown in sections 2 and 3, in general there is no chance to find a parse covering the whole utterance in every case. Information about the order of the semantic units is needed for processing of some of the spontaneous effects, like self repairs.
within the user utterances. This kind of dynamic adaptation of the confirmation strategy shows the users that the system has problems or recovers from trouble in understanding the user.

**Initiative strategies** have already been discussed in section 4.1: a sophisticated system is supposed to perform a mixed initiative dialog. Nevertheless, the initiative strategies might be changed dynamically according to the current understanding performance, too. A conversational system is expected to guide the dialog when the user is not doing so. An active user leading the conversation should not be restricted. Thus, the initiative strategy has to be adapted to the user according to his abilities.

**Formulation of system utterance** is well known to affect the users behavior. Given the same informational content, different wordings can make the system look smart or dumb. Even the quality of the synthesized speech affects the users utterances, both in content and in appearance: some users tend to mimic the systems utterances using the same words and the same prosody. A lot of these effects are already reported from WOZ experiments [11, 23].

Currently, there are no sufficient examinations of the effect of the different strategies. However, with the number of demonstration systems the corpora of spoken human machine dialogs is growing rapidly. A systematic variation of the strategies outlined above is worth to be performed. This will lead to much better models of real users and their behavior.

In order to model the system reaction, there seems to be agreement in the community to use **dialog acts** [6] to describe the users and systems intentions. In a very simple interpretation system it is sufficient to extract the parameters of each utterance that are relevant for the task. When a system evolves towards conversational capabilities, the representation of conversational intentions benefits from the usage of dialog acts. However, there is still no commonly agreed definition of the term dialog act [8, 26, 31, 22]. While we would appreciate a proper definition we still doubt that a comprehensive and complete list of dialog acts is possible and would be accepted by everyone. There are always excuses to use a different ontology or methodology.

On the other hand there seems to be agreement that the dialog planning process is determined by the most recent user utterance, the dialog state (i.e. all user and system utterances of the cur-

---

3 According to [1] the meaning is something that one wishes to convey, esp. by language.
rent dialog), and the static strategy parameters. Therefore we can see the dialog state as a discrete point in the space of possible dialogs, and the generation of an system utterance is a transition in this dialog space. It is the goal of a dialog step function DSTEP to specify the subsequent dialog state for each particular point in the dialog space. The DSTEP function represents the systems dialog model, and it contains the effect of the dynamic strategies. Again, there is no common agreement on how to implement this transition function.

5 Evaluation Methodology

After building a spoken dialog system, we want to find a rating whether it is a good or a bad system. There are two principal approaches to system evaluation [29]: the black box evaluation methodology only considers input output behavior of the whole system, whereas in the glass box evaluation the intermediate results of modules are analyzed.

When analyzing the systems behavior, the crucial problem is that there is no single "reference" dialog. Judging the appropriateness of a system utterance has to be performed manually by a referee. This independent expert has to provide an annotation of each system utterance in the context of the current dialog. There could not be a reference answer since several different system reactions might "make sense" and are permitted as proper system utterances. When the annotation of each dialog is performed, the corresponding rating of appropriate system reactions can be calculated automatically. The next and most important measure for system evaluation is the resulting dialog success rate, i.e. finding out whether the users general request was satisfied. Finally, a dialog is supposed to be better when it was performed faster, both in the number of turns and the time elapsed to get the information. Unfortunately, these measures differ largely for different domains and tasks. Thus, a comparison of different systems is not easy to accomplish. First approaches to standardized system evaluation are reported in [29, 12, 16].

Evaluation of single components requires access to internal protocols of the dialog system, e.g. at module interfaces. As described above, the typical result of a word recognizer is a word string or word graph, the result of a parser is a semantic description. Since we want to deal with larger corpora of data, we prefer to have an automatic method to calculate the performance of a module. Apart from the dialog manager the other modules can be evaluated by comparing their actual result against a reference result, i.e. for the word recognizer we need the transliteration and for the parser we need a semantic annotation of the users speech. As argued above, the dialog manager could not be evaluated by comparing the system utterance with some reference utterance. For the recognizer and the parser this approach is feasible and leads to ratings that could be compared with other systems.

Word Accuracy (WA) is a widely accepted evaluation measure for word recognizers. The automatic calculation of WA for a given set of recognition results requires the existence of reference transliterations for all spoken utterances. The reference answers consist of a transcription of what was actually spoken. WA is calculated as a percentage using the formula

$$WA = 100 \left(1 - \frac{W_s + W_I + W_D}{W}\right) \%$$  \hspace{1cm} (1)

where $W$ is the total number of words in the transliteration, and $W_s$, $W_I$, $W_D$ are the number of reference words which were substituted, inserted, and deleted in the recognized string, respectively. This measure is easily extended to rate the accuracy of word graphs considering their density.

Accordingly, we define the quality of a parser by calculating the semantic concept accuracy (CA) which considers only the information content represented by semantic units (SU):

$$CA = 100 \left(1 - \frac{SU_s + SU_I + SU_D}{SU}\right) \%$$  \hspace{1cm} (2)

where the semantic units are attribute-value pairs that are present in the semantic annotation. The substitutions, insertions, and deletions are counted in analogy to (1). The definition of the attributes relevant for understanding is determined by domain dependent task parameters which reflect the functionality of the system, and by dialog control markers for words and phrases like yes, no, good morning, could you repeat etc.

As an intermediate result we can calculate the coverage of the parser by measuring the semantic concept accuracy obtained on the transliteration, i.e. assuming to have a perfect word recognizer. This gives an indicator of the parsers ability to deal with phenomena of spontaneous speech.4

4Interestingly, the parser does not need to find correct parses for all utterances. Actually there might be parts which could not be parsed. If these parts do not contain
6 A Spoken Dialog System

In order to obtain a measure for the understanding accuracy, the CA is calculated when feeding actual recognizer output into the parser. Thus, the chain of word recognizer and parser are evaluated as a whole which is a step further towards black box evaluation of a system. As a side effect the comparison between WA, coverage and the obtained understanding accuracy shows the robustness of the parser against recognition errors.

Utilizing the reference transliterations and the semantic annotations all these measures can be calculated in an automatic way. Figure 2 shows this procedure, where the program \texttt{eval\_seg} is used to calculate the Levenshtein distance. A mapping procedure might be necessary to perform format conversion between the parser output and the semantic annotation format.

At Erlangen University we have installed a train time table information system (TII) which is able to answer inquiries about German intercity train connections. This system is shown in Figure 3 and contains the three major modules word recognizer, parser and dialog manager. This system is connected to the public telephone network and was used to collect dialogs with naive users. A first collection of about 1000 dialogs was used as a test corpus to evaluate the system and the modules (cf. [14, 4]). Table 2 gives an overview of the test corpus.

A semantic annotation scheme was developed containing the task parameter names and the dialog control marker as attributes. Together with their corresponding values they make up the semantic units for equation (2). Additional attributes had to be introduced for underspecified (elliptic) utterances, e.g. when replying a sole city name to a question on the location of departure. Disambiguation is performed by the dialog manager using the dialog context, so the semantic annotation needs to represent this ambiguity. Thus we obtained a total of 38 different classes of semantic units.

| Total number of dialogues | 1092 |
| Total number of utterances | 10114 |
| Total number of words     | 33477 |
| Average of words per utterance | 33.31 |
| Total number of semantic units | 14584 |
| Different classes of semantic units | 38 |

Table 2: Figures of the test corpus.

Discussions arose when calculating the semantic concept accuracy since the semantic units consist of the pair of a parameter and its value. Lets
say, for example, the user had uttered

Ich möchte morgen nach Bonn
(I want to go to Bonn tomorrow) (3)

and the correct semantic annotation consists of
the two SUs

\[ \text{[goalcity: Bonn, date: tomorrow]} \]. (4)

A substitution of morgen (tomorrow) with the
word morgens (in the morning) results in the
semantic units

\[ \text{[goalcity: Bonn, partofday: morning]} \] (5)

leading to a misunderstanding of both the semantic
concept and its value. On the other hand the
substitution of Bonn with Berlin would result in

\[ \text{[goalcity: Berlin, date: tomorrow]} \] (6)

with only the value of the parameter goalcity
being misunderstood. One could argue that the
latter case is more severe than the previous, but
the definition (2) judges both as equal\(^2\).

Since we consider both the parameter name and
the value as properties of the semantic unit,
the whole unit must be recognized correctly. A
quick comparison with possible word recognizer
errors shows, that (2) is an appropriate measure.
When counting the word errors we do not consider
homophones to be less severe (e.g. I look in your
[eyes | ice]), and we do not consider a mismatch in
tense or gender as a less severe error. All of them
are just wrong. The calculation of the semantic
concept accuracy is performed in analogy resulting
in an error no matter how close the result is.

First results using this dialog corpus have already
been reported in [14]. In [13] we found that,
while the system evolved, 53.1% of all dia-
logs were finished successfully. An average dia-
log took 154 seconds of connection time and con-
tained 9.2 user utterances. Using this corpus with
our current word recognizer, we obtained a WA of
79.1% [18]. The parser has a coverage of 92.8% on
the transcriptions of spontaneous speech. The
sequence of word recognizer and parser results in
a CA of 79.8% [4]. We found that in our case
the relation between WA and CA is nearly lin-
ear, which means that the word recognizer and
the parser are well matched.

7 SUMMARY

In this paper we tackled the difficulties of defin-
ing the term understanding. We presented a set
of theses which we think are worth to be consid-
ered when building a speech understanding sys-
tem. While robustness against phenomena of
spontaneous speech might be modeled explicitly,
we favor the approach of generating partial de-
scriptions. Three different definitions of the term
dialog were presented and only the conversational
system was found challenging for further research.
Final steps of understanding an utterance are the
anchoring within a dialog context and the context-
tual interpretation utilizing world knowledge. A
dialog control mechanism which specifies the sys-
tem reaction is seen as the application of a dialog
step function. In this model, every dialog state is
represented as a discrete point within a space of
possible dialogs. Dialog strategies, e.g. regarding
the confirmation or dialog initiative, are repre-
sented as parameters of the dialog step function.
Experiments have shown that the actual wording
of the system utterance is an important strategy
parameter, too. For evaluation of understanding
systems, we presented the measure semantic con-
cept accuracy which is calculated in analogy to
word accuracy. A short description of our own
spoken dialog system, some effects observed, and
the resulting figures complete this paper.

8 ACKNOWLEDGEMENTS

We are very grateful to our colleagues at the Uni-
versity and at FORWISS, Erlangen. Parts of
the research presented in this paper are funded
by the Daimler-Benz Research Institute, Ulm,
in the project SYSLID. Parts of this work were
supported by the German Research Foundation
(DFG) under contract number 810 830-0.

REFERENCES

[10], pages 125–128.
[3] A. Baekgaard et al. The Danish Spoken Lan-
guage Dialogue Project — A General Overview.
In Dalsgaard et al. [10], pages 89–92.
[4] M. Boros et al. Towards understanding sponta-
eneous speech: Word accuracy vs. Concept accu-
ricy. In Proc. Int. Conf. on Spoken Language
[5] A. Brietzmann et al. Integration of Acoustics-
linguistics for a Robust Speech Dialogue System.


Some Ideas for the Automatic Acquisition of Dialogue Structure

Jan Alexandersson
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken
alexandersson@dfki.uni-sb.de

ABSTRACT

We are reporting on some initial results on the automatic acquisition of plan operators for a plan recognizer. The operators are derived from the VERBMOBIL corpus of negotiation dialogues hand-annotated with dialogue acts. The corpus is pre-classified and a set of plan operators is derived for every class. The plan operators are then tested on a set of unseen data. We also show some initial results.

1 Introduction

The VERBMOBIL project [13, 5] is a long term project founded by the German ministry for Education, Science, Research, and Technology, for developing an automatic interpreting system for taks oriented face-to-face dialogues. The main work so far has been concerned with the translation of natural spoken German into English, in the domain of appointment scheduling.

The project involves 31 industrial and academic partners concerned with a broad range of activities spanning from data collection, speech recognition, machine translation to pure system integration. The current status of the working system consists, from the dialogue module's point of view, of two types of processing approaches - deep and shallow. Both lines use the output from the speech recognition components (speech recognition and prosody) as input. The deep processing line uses a more traditional linguistic approach consisting of a syntax-semantic parser, a transfer module, and a tactical generator. The shallow line uses message extraction techniques to analyse, transfer, and generate. The semantic evaluation component, and the dialogue component are concerned with providing contextual information and resolving transfer relevant ambiguities.

The dialogue component of the VERBMOBIL system [1] consists of 3 main components.

The statistical component providing different modules in the system with top-down predictions.

The dialogue memory consisting of three structures, the dialogue sequence memory, the thematical structure, and the intentional structure.

The plan recognizer constructing, for instance, the intentional structure.

The dialogue processing is centered around dialogue acts [4], which are used as basic entities for both the prediction process as well as the dialogue sequence memory, and the plan recognizer.

This paper describes some ideas how one can utilize a hand-annotated corpus for automatic derivation of plan operators for a plan recognizer. We show how these ideas enable us to overcome some problems we experienced when we tried to hand code the operators. The paper is structured as follows. In section 2 the intentional structure is presented. A short introduction to our dialogue act hierarchy, and a survey of the plan recognizer is given. Section 3 gives some hints on what problems we are faced with when we want to build the structure. We show how the corpus was used in section 4. We present some of the results in section 5, and conclude the paper in section 6.

*This work was funded by the German Federal Ministry for Research and Technology (BMBF) in the framework of the VERBMOBIL Project under Grant 01IV101K/1. The responsibility for the contents of this study lies with the authors. The author wishes to thank Elisabeth Maier and Norbert Reithinger for comments on earlier drafts of this paper, and Ralf Engel for his help with the implementation.
2 The Intentional Structure

From our perspective, dialogue acts represent the intended meaning of an utterance and abstract over the possible linguistic realizations. We have defined a set of 42 domain dependent as well as domain independent dialogue acts [4]. Some of them are purely illocutionary, e.g. requesting a proposal for a date or requesting a comment on a proposal, but they can also comprise propositional content, e.g. proposing a date or giving a reason for rejecting a proposed date. Traditional illocutionary acts as they were proposed by Austin [3] and later integrated into Searle’s theory of speech acts [9] aim at a rather coarse-grained typology detached from their propositional content.

The intentional structure is a tree-like structure mirroring different abstract levels of the intentions of the dialogue. It is built of four levels (see fig 2).

The Dialogue Act Level implements,
with some minor extensions, the dialogue act hierarchy.

The Turn Level connects the utterances inside a turn.

The Phase Level distinguishes the three dialogue phases greeting phase, negotiation phase and closing phase.

The Dialogue Level spans over the whole dialogue, eventually distinguishing more negotiations.

The reasons for dividing the structure are.

- The dialogue act hierarchy is used as a way of abstraction from the actual topic – it should not matter whether we negotiate a time or a place.

- We are interested in reduction, i.e. condensing the turn to its important, information carrying parts. The turns level is also a convenient way of making the plan recognizer robust against gaps (see section 3) in the input.

- The phases must be computed because of translation purposes – translation ambiguities can be resolved with dialogue phase information.

The turn level is the most problematic layer, and the aim of this work is to explore some ideas about how one can derive the plan operators describing the structures of turns automatically from an annotated corpus.

Building the Intentional Structure

There are a lot of similarities between plan recognition and parsing as pointed out by Vilain [12].
There are many advantages to choosing a parsing approach since the area is quite explored and there are a lot of efficient algorithms described in the literature. We have chosen a top-down left-to-right parsing algorithm with destructive backtracking to build the intentional structure. There are several reasons for that. Since we know that the dialogue is about negotiation, we make use of this knowledge. The plan operators also maintain an internal context which is “carried around” in the tree as it is constructed.

**Plan Operators**

Our plan operators follow the usual paradigm and decompose a goal into one or more subgoals. Pre-actions and actions can be defined which are evaluated when the operator is inserted in the tree and when all its subgoals are completed. It is also possible to define constraints to an operator which prevent it from being applied in certain contexts. An example of a plan-operator dividing the goal \texttt{DOMAIN-DEPENDENT SUGGEST \textquestionmark{INCONTEXT} \textquestionmark{OUTCONTEXT}} into an operator processing iterations is shown below. The operator sets the last important utterance turn to suggest when completed, and modifies the \texttt{\textquestionmark{OUTCONTEXT}} at the same time.

```
begin-plan-operator SUGGEST-OPERATOR
pre-actions (nop)
undo-pre-actions (nop)
goal [DOMAIN-DEPENDENT
  SUGGEST
  \textquestionmark{INCONTEXT}
  \textquestionmark{OUTCONTEXT}]
subgoals (sequence
  [IN-DOMAIN-DEPENDENT
    SUGGEST-ITER
    \textquestionmark{INCONTEXT}
    \textquestionmark{OUTCONTEXT}])
actions (set-last-important-dialogue-act
  'suggest
  \textquestionmark{INCONTEXT}
  \textquestionmark{OUTCONTEXT})
undo-actions
  (unset-last-important-dialogue-act)
end-plan-operator
```

There are some special keywords like “sequence” used in the operator above. The operational semantic for this is: the subgoal(s) has (have) to be solved sequentially. It is also possible to use the key words “xor”, “and”, “iterate” and “optional” with their obvious operational semantics. The symbols beginning with a question mark are variables.

The plan recognizer operates on two single structures, the dialogue memory [6, 7], and the sequence structure. We do not, for efficiency reasons, keep multiple contexts so in the process of defining the plan operators we have to define undo-operations for both the “actions” and the “pre-actions” to reset the side effects performed.
by the actions/pre-actions when the intentional structure is backtracked away.

To increase the run time efficiency, we use two standard optimizations. The algorithm makes use of a tabulation mechanism, which stores a copy of the complete substructures that would be destroyed in case of backtracking. The copy can then be inserted when the recognizer tries to solve the same goal again. We also compile out so-called reachable constraints which prevents an operator to be applied when there is no possibility for its successful application. This has turned out to be a very efficient, not to say necessary, tool for improving run time performance.

Repair

Exploring our first approach [2] we had to extend the plan recognizer to process unexpected input — repair. This is done by means of a set of specialized repair-operators together with some additional methods for inserting them into the tree. Dialogue acts like GIVE_REASON and CLARIFY_STATE are of a property appearing everywhere in the dialogues. Instead of writing an extra operator handling this, we use this repair mechanism for “repairing in” these dialogue acts.

A special problem are meta dialogues in the dialogue like clarification dialogues. They occur (at least) two forms — pure clarification dialogues where one speaker makes a single CLARIFY_QUERY and the other speaker optionally responds with a single CLARIFY_ANSWER back. Unfortunately this is rare, and instead the clarification dialogues are embedded in the turns, which makes it very hard to mirror in the structure. Currently we process clarification dialogues by means of repair.

Problems with our first approach

It is a well known fact that man-machine dialogues and dialogues between two people that are allowed to speak freely, even in task oriented dialogues, differ in structure and, more importantly and not in favour to us, regularity. Two persons trying to convey the same goal, manage in our corpus to speak in very different ways, which makes the task of hand coding this behaviour very hard or even impossible. Our first version of the plan recognizer was based on hand coded plan-operators, and as an effect of the very different structures of the dialogues we processed, a lot of repair was needed. This in turn made the structure sometimes so awkward, that it was hard to observe any meaning from the structures.

3 Real World Data

Robustness

We here indicate some problems with the input to the dialogue module. Besides the well-known problem with speech recognition, we can observe the following problems:

Wrong segmentation A big problem with analysing spoken input is segmentation. Since a turn consists of not just one utterance, the turn has to be split into pieces. This process would be fairly simple if it was not for the fact that people do not speak grammatically correct, and, as pointed out above, problems with the speech recognition.

Missing deep information When the deep processing components fail to analyse, transfer or generate the turn, shallow methods are being used. These methods do not provide any deep information, but the dialogue act.

Multiple Dialogue acts The components analysing utterances in the VERBMOL system are currently just able to attach one dialogue act per utterance, although the utterance has more than just one function. This could make a machinery based on a hand-coded dialogue model fail.

To demonstrate what problems we are facing, we have taken an example from our corpus to show what consequences a missing multiple dialogue act can have:

Example 3.1

PS1001a: ja prim, (FEEDBACK)
(Well)

1This corresponds to the improvement of a parser called “Top-down parser with Bottom-up filtering” as described in [14].

2With a lexicon size of 3000 fullforms, the speech recognition components in the running system have a word accuracy of about 75%. This means that the actually spoken sentence is more or less guaranteed not to be found in the word lattice.

3The translation of the examples is not produced by VERBMOL.
PS1001b: dann lassen Sie uns doch noch einen <eins> Termin ausmachen.
(INIT.DATE)
(LET US MAKE A DATE)

PS1001c: wann wär's Ihnen denn recht?
REQUEST_SUGGEST_DATE
(WHEN DOES IT SUIT YOU?)

BS1002a: also ich dachte noch in der nächsten Woche, auf jeden Fall noch im April <Klicken>.
(SUGGEST_SUPPORT_DATE)
(WELL I THOUGHT IN THE NEXT WEEK, IN ANY CASE IN APRIL)

PS1003a: #Klicken# ja am Dienstag den sechsten April hätten ich noch einen Termin frei allerdings nur nachmittags.
(ACCEPT_DATE)
:YES ON TUESDAY THE SIXTH OF APRIL IS POSSIBLE FOR ME, BUT ONLY IN THE AFTERNOON)

PS1003b: geht es da bei Ihnen <lIhnen> auch <Klicken>?
REQUEST_COMMENT_DATE
(DOES THAT SUIT YOU?)

...  

Here the utterance PS1003a\(^4\) is annotated with just ACCEPT_DATE, but it also has the function of proposing a new date (SUGGEST_SUPPORT_DATE). The following utterance PS1003b annotated with REQUEST_COMMENT_DATE can not, according to our dialogue model, follow an utterance of type ACCEPT – one should introduce a new topic before one can request a comment on, in this case, a date. There are different ways of coping with this, and as we will see, the approach proposed in this paper can contribute to handle these cases (see also [7][8]).

4 Using an Annotated Corpus

Because of the problems with hand coded plan operators, we were looking for alternative approaches, and since we have quite a big corpus of negotiation dialogues we were interested if it was possible to make use of it by automatically deriving plan operators from it. In this case, could they be used in the running system?

The VERRMOBIL corpus consists of 12 CD-Roms of transcribed German-German as well as German-English negotiation dialogues of which 5 are hand-annotated with dialogue acts. Currently, no time information has been annotated in the corpus. For the experiments described in this paper, we have used 3 CD-Roms of German-German dialogues, containing 276 dialogues for training and 50 German-English for testing.

Preparing the Corpus

The idea when deriving the plan operators is to classify the turns and then derive a set of operators for each class. We use a small amount of knowledge about the dialogue acts and the structure of the dialogues, which resulted in 9 classes (see below). Utilizing the knowledge about the dialogue acts, we first prepare the corpus. This is done in three steps:

1. First filter out parts of, or even utterances which do not contribute to the actual negotiation.

Utterances which lie outside the scenario are attached with dialogue acts like GARBAGE, REFERENCE_SETTING or DIGRESS_SCENARIO. Also some dialogue act types, which appear anywhere in the dialogues, like GIVE_REASON or CLARIFY_STATE, were filtered out from the utterances. They will be processed by means of the repair mechanism. The exceptions are turns consisting of just a CLARIFY_QUERY or a CLARIFY_ANSWER. They are detected and processed by hand-coded operators.

2. The next step is to make use of our dialogue act hierarchy, and map the more special dialogue acts to the more general ones (cf. SUGGEST_SUPPORT_DATE is mapped to SUGGEST)

3. Finally we “collapse” iterative appearances of the same dialogue act to just one. This means that for instance a turn consisting of two utterances annotated with SUGGEST_SUPPORT_DATE and SUGGEST_EXCLUDE_DATE respectively is in the first abstraction step mapped to SUGGEST SUGGEST, which is is reduced to just SUGGEST. In this process we notice which dialogue acts appear more than once in a row, and correspondingly, we have to modify the set of dialogue acts implementing the dialogue act hierarchy.

\(^4\)Speaker A is indicated by PSn, and speaker B by BSn
Example 4.1

- Original version

MM4002a: <P> <A> ich finde es zwar auch ganz schön
(FEEDBACK_ACKNOWLEDGEMENT)
(I find it also very nice)

MM4002b: +/was da/ <P> <hrm> daß wir da beim <P> <A> Sport unsere
<uunsere> Besprechungen <b>Besprechung> gehalten haben <i>ham>
(DIGRESS_SCENARIO)
(what there... that we talked during the last exercise)

MM4002c: aber <A> ich finde doch daß wir auf die normalen <A> Werkstage
zurückgreifen sollten
(SUGGEST_SUPPORT_DATE)
(But I think we should aim for the regular working days)

MM4002d: und uns das Wochenende für
die Familie vorbehalten sollen,
(SUGGEST_EXCLUDE_DATE)
(and reserve the weekend for the family)

MM4002e: <A> wie sieht es da <P>
bei Ihnen bezüglich einer Besprechung
montags denn <i>enn> aus <#Störgeräusch>
<P> (SUGGEST_SUPPORT_DATE)
(What do you think about a meeting on Monday?)

Filtered version

MM4002: aber <A> ich finde doch daß
wir auf die normalen <A> Werkstage
zurückgreifen sollten und uns das
Wochenende für die Familie vorbehalten sollen , wie sieht es da <P> bei Ihnen
bezüglich einer Besprechung montags denn
<i>enn> aus <#Störgeräusch> . <P>
(SUGGEST)
(But I think we should aim for the regular working days and reserve the weekend for the family.
What do you think about a meeting on Monday?)

Deriving Plan Operators from the corpus

The next step in the process of deriving plan operators from the corpus is to classify the turns. Examining a part of the corpus, we found that most of the turns fit into one of the following 9 classes: `greet`, `greet-initiative`, `bye`, `response`, `response-initiative`, `initiative`, `initiative-response`, `unknown`, and `misc`. In figure 3 we see the classes and their corresponding dialogue acts. The class `initiative-response` is worth a comment – we found 12 turns where the speaker suggested something and rejected or accepted the suggestion herself.

When dividing the corpus we respected all turn patterns occurring more than once in the corpus.

The Learning Algorithm

For deriving the plan operators we have used the Bogus system [10, 11]. It is a system for deriving, for instance, Hidden Markov Models and Stochastic Context Free Grammars. It is based on an approach to the learning problem of probabilistic language models, known as “Bayesian Model Merging”, and has some properties we want to utilize. Since we are interested in real time performance we can, during runtime, take advantage of statistic information generated by the system. An example of the result derived by the system is given in the appendix.

5 Evaluation

Overall Evaluation

We tested the derived plan operators on 50 dialogues with the following results: From 531 turns in total, we managed to classify 471 (87.7%) turns. Of these 471 two thirds (67.2%) were in the training corpus and correctly classified. Of the remaining 155 (32.8%), 93 were correctly classified, and 62 were incorrectly classified. This means that a total of 409 of the 471 (86.8%) was correctly classified. Of the remaining 62, about 50% were involved with repair.

Detailed Example

We now take a look at a running example taken from one of the test dialogues. We show how the plan recognizer incrementally builds a structure given the following turn:

5It has, as pointed out in the outlook, to be further investigated if these classes are sufficient.
Figure 3: The 9 classes and their corresponding dialogue acts

Example 5.1

THW002a: das <P> ist zu knapp
(REJECT_DATE)
(This is not enough time)

THW002b: weil ich <P> <:<#> ab:
dem dritten in Kaiserslautern bin
(GIVE_REASON)
(because I will be in Kaiserslautern from the third)

THW002c: <:#> genaugenommen nur am
dritten (SUGGEST_EXCLUDE_DATE)
(to be precise just on the third)

THW002d: <A> wie wäre es denn <P>
am<2> <P> +/-/=+/ <ahm> <:<#> Samstag, 
den> zehnten Februar?
(SUGGEST_SUPPORT_DATE)
(How about on... - ehm - Saturday the tenth of February)

In figure 4 we see a screen-dump of the intentional turn structure after processing the first two utterances. The recognizer assumes that the turn is a response-turn (OPERATOR-S-RESPONSE-5). The GIVE_REASON is repaired into the existing tree. The operators with -ATOMIC- in their names are unified with the information in the dialogue memory, whereas the operators with -ITER- are processing iterative occurrences of the same subgoal.

Figure 5 shows the structure after the third utterance, SUGGEST_EXCLUDE_DATE, is processed. The top goal of the turn has changed into a response-initiative. Observe that the

![Diagram](image)

Figure 4: After reject_date and give_reason

GIVE_REASON now is attached to the SUGGEST_EXCLUDE_DATE - it is impossible without information about the propositional content of the utterances to determine where it should be attached, so currently this type of repair is done in a kind of ad hoc manner.

Finally, the second suggestion SUGGEST_SUPPORT_DATE is added to the tree to the topmost domain-dependent suggest-operator (see figure 6). This operator was modified to handle iterative appearances of utterances with the same dialogue act.

6 Outlook

We have presented a method for the semi-automatically derivation of plan operators from a
hand-annotated corpus of dialogue act sequences. The results of the first evaluation has yielded quite promising results. We will continue exploring these ideas. There are, however, some tasks which need more work:

- The derived plan operators do not cover 100% of the test corpus. Should this be taken care of by more sophisticated repair methods?

- Are the classes optimally chosen? — should we look at subclasses and build some classes on top of them. It is tempting to divide the classes concerned with response and initiative into subclasses and then use them to construct the classes response-initiative and initiative-response based on them.

Moreover, the classes chosen are a little bit to coarse-grained. They do not, for instance, respect giving the initiative away (REQUEST:SUGGEST:DATE), which for instance could make it possible to resolve some of the turns in the category "unknown".

- How big must the training set be before we get a sufficient coverage?

- To even more speed up the analysis process, we can use the prediction mechanism provided by the statistical component, and predict what class(es) is (are) most probable to come next.

This could also be used to predict language models for the speech recognition components to improve the recognition rates.

- Is the learning system suitable for our purposes? One advantage and at the same time risk with this approach is that the derived grammar might overgenerate, i.e. it is possible to recognize input which is not in the training set. A good guess is that a grammar like the one yielded by the BOGUS system is a more compact and efficient representation than just the naive grammar constructed by letting the top symbol of the grammar expand into exactly every sentence in the training set.

References

in the cmp-lg electronic archive under no. cmp-lg-9502008.


Appendix

We here give an example of a class and its corresponding grammar. The number means that the pattern occurred that many times, and the parenthesized symbols means that the symbol occurred more than once in a row.

:: data greet-initiative

(5 (introduce_name) (suggest))
(4 (greet) (suggest))
(3 greet (introduce_name) (suggest) request_comment)
(3 greet (introduce_name) (suggest) request_suggest)
(3 (greet) (init) suggest request_comment)
(3 greet (introduce_name) init (suggest) request_comment)
(3 greet (introduce_name) init (suggest))
(3 (greet) (init))
(2 greet (introduce_name) init suggest request_suggest)
(2 introduce_react introduce_name (suggest))
(2 (greet) init request_suggest)
(2 greet (introduce_name) motivate_appointment init suggest)
(2 greet (request_suggest))
(2 (greet) motivate_appointment (init))
(2 greet introduce_name (accept) suggest)
(2 (greet) init clarify_query)
(2 greet (introduce_name) init motivate_appointment)
(2 (greet) (introduce_name) motivate_appointment init request_suggest)
(2 introduce_name motivate_appointment (suggest) request_comment)
(2 greet introduce_name motivate_appointment (request_suggest))
(2 (greet) introduce_name motivate_appointment (suggest))
(2 (introduce_name) clarify_query)
(2 (introduce_name) init suggest)
(2 (greet) introduce_name request_suggest)
(2 (greet) introduce_name reject (suggest))
(2 (greet) (introduce_name) init request_suggest)
(2 (greet) introduce_name clarify_query)
(2 (greet) init)

For the grammar, the leftmost position of the right hand side of a rule is a tuple which should be read probabilistic-n-of-occurrences – this information can be used during runtime to choose the operator with the highest probability.

:: grammar greet-initiative

(0 -> (1.536e-2;2.00000 1 36))
(1.536e-2;2.00000 1 59 2 36)
(1.536e-2;2.00000 1 69 2)
(1.280e-3;1.00000 1 69 59 2)
(4.353e-2;4.00000 58 36)
(7.170e-2;5.00000 64)
(4.353e-2;4.00000 65 2)
COOPERATIVE RESPONSE PLANNING IN CDM: REASONING ABOUT COMMUNICATIVE STRATEGIES

Kristiina Jokinen
Computational Linguistics Laboratory
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara-ken
6301-01 JAPAN
kris@is.aist-nara.ac.jp

Abstract

This paper discusses the problem of generating communicatively adequate responses in information seeking dialogues and advocates the Constructive Dialogue Management (CDM) approach, which is an advance on previous research in that communicative principles are explicitly taken into consideration. Communicative principles are general inference rules which describe rational, cooperative action in establishing sufficient mutual knowledge for the fulfilment of the task. With their help, the system can reason about appropriate communicative strategies and overcome shortcomings in its knowledge base. The principles also allow planning of responses in different communicative situations, supporting flexibility and generality of dialogue management.

1 Introduction

Early work on communicatively adequate responses was concerned with specifying the system’s input and output conditions so that the system’s overall behavior would exhibit features that could be called cooperative. For instance, if the user request has an unsuccessful result, the system would give additional information, attempt to correct misconceptions [14, 15], and provide the user with extra helpful information in anticipation of further questions [18, 1]. Communication was seen as a ‘one-shot’ process whereby all the pertinent information was given to the user at once. The work by [16] liberated the system from one-shot responses: dialogues are now understood as negotiations, and the system’s reasoning capabilities are extended so that the user’s follow-up questions can be analyzed and an appropriate explanation be given that suits the user’s expertise level. However, the system is still assumed to be naively benevolent in that it adopts the user’s goal and produces appropriate responses within the given task: its communicative skills are limited and it gets stuck if the input is unknown or somehow deficient, neither can it reason about the best way to organise its goals if a conflict occurs, cf. [9]. Important aspects of successful communication deal with the ways in which the participants clear up vagueness, misunderstandings, or lack of understanding in their contributions. Communication usually arises from non-linguistic goals (rent a car, repair a pump, reproduce the partner’s route on a slightly different map, etc.): the distribution of relevant knowledge is such that the participants cannot reach their goals alone, so they need to collaborate with each other to obtain the missing information, cf. [10]. The success of the task is dependent on the speakers’ ability to establish sufficient mutual knowledge for fulfilment of the task, and in this, different strategies that the speakers use in managing mutual knowledge prove more successful than others: it is not enough that the participants convey appropriate factual information, but the better they can discover pertinent mismatches and recover from problems, the more accurately and quickly the task is completed.

The Constructive Dialogue Model (CDM) reported in this paper is an advance on the previous research in that communicative principles are explicitly taken into consideration. With the help of communicative principles, the system can plan responses which are communicatively appropriate in different communicative situations, independently of whether the situations concern the same domain tasks or not. The system thus reasons about appropriate communicative strategies: it chooses among different continuation possibilities (give more information, suggest another alter-
native, repeat the goal etc.), and can thus overcome shortcomings in its knowledge base.

Reference to general communicative principles may seem abstract from the point of view of practical NLP applications, but it is believed that the functionality of systems can be significantly improved only if the design decisions can rely on an adequate theoretical basis. A mere extension of contextual knowledge is not enough: firstly, a huge amount of contextual knowledge is required, and secondly, the type and nature of contextual knowledge is difficult to determine. Instead, the categorisation of contextual information by referring to general enablers and requirements for communication allows us to subsume different cooperativeness features into a few communicative principles, and thus provide a uniform basis on which content planning can take place.

This paper discusses communicative principles and their use in strategic planning, and presents the CDM model as a new approach to dialogue management. The paper is organised as follows. Section 2 provides the empirical basis of the research and identifies the problems which exemplify the theoretical point: how the dialogue participants make strategic choices in planning communicatively adequate responses. Section 3 introduces the concepts underlying the Constructive Dialogue Model, and Section 4 describes the implemented CDM system. Section 5 presents conclusions and future work.

2 COMMUNICATIVE STRATEGIES

We call the ways in which mutual knowledge is established, maintained, modified and exploited communicative strategies. They are distinguished from problem-solving strategies which are task-dependent means to resolve the task at hand. Communicative strategies are based on the agent's rationality: actions are chosen so as to conform to the shared assumptions about cooperative behaviour in a given situation.

In the following we discuss communicative strategies in the light of corpus examples, and pay attention to the inferences that the participants draw when they plan contributions that clear up vagueness, knowledge limits and conflicts.

2.1 CLEARING UP VAGUENESS

Vague contributions have unclear content in that they contain concepts which are not understood or which cannot be connected to the dialogue situation. Hence, clarification is needed. In fact, the crucial feature of dialogues is that contributions always seem to contain vagueness on some level of interpretation, and the dialogues thus resemble negotiations rather than straightforward question-answer sequences. A common ground is gradually built up over several contributions (see [7]), rather than in a 'one-shot' process where all the information is given to the partner at once.

Consider the following dialogue, in which vagueness appears twice in User1 and User2:

(1) User1: hello can you tell me where there are some good restaurants
Wizard1: The YP has no details of the quality of restaurants. Can you be more specific as to area and/or type?
User2: I'm looking for some spicy food in the centre of manchester
Wizard2: I have Indian, Chinese, Thai, and Mexican restaurants listed.
User3: what about an indian then
Wizard3: Please wait...
Here's the list: <>

The user has supplied the keyword "restaurant", so the wizard is able to interpret the request with respect to the current task and application (provide information about restaurants). At the beginning of the dialogue, the evaluation of the user goal leads the wizard to a 'new request' strategy, and the user's goal of finding restaurant information is adopted. Filtering this goal via the application back-end, the wizard finds that there are many restaurants and it is uncooperative to give the user the information about all restaurants. Thus the wizard formulates her own goal of restricting the database search with the help of the restaurant type or location. Although the user has supplied a modifying adjective 'good', this does not help, since 'good' does not narrow down the number of restaurants. The evaluation of the user contribution thus results in two interrelated intentions: the wizard wants to provide the user with restaurant information, and she wants to restrict the database search.

The user is most probably ignorant of the specification parameters that can be used in the database search, so the wizard shows consideration towards the partner and asks especially about the type and location instead of an open, general question Can you please specify? (to which the quality adjective "good" would also be a possible answer). More-
over, the wizard adds an explanation about why the requested information could not be given in the first go (there is no information on the quality of the restaurants in the Yellow Pages). Hence, instead of simply informing the user of a failed search (The YP has no details of the quality of restaurants), or only advancing her own goal (Can you please specify), the wizard appears maximally cooperative, and explicitly expresses all the information that she thinks is relevant in a successful completion of the task.

Neither of the cooperativeness features is necessary though. Depending on the speaker’s conversational posture and risk-taking ability ([6]), some part of the information content can be ignored or omitted, in order to contribute to conversational fluency, to achieve a particular communicative effect, to maintain one’s ‘face’, etc. If the wizard had been bolder, she could have continued the dialogue with the open general question only, without the explanation, or without the type/location specification. A response with only an explanation would sound uncooperative in this situation, however, since it would leave the user to cope with the wizard’s knowledge problems on her own; such a response assumes that the speakers already share the topic and the knowledge about how to cope with the explanation and continue the topic.

In her next contribution the user provides the location of the restaurant, and also the food type, although the type specification seems not to be understood (a misspell and vague adjective). Because the type has already been asked, the wizard prefers to give the alternatives for the user to choose from, instead of explicitly asking for the type again. The response is communicatively more adequate than a simple repetition of the type question (What type of food/restaurant would you like?), since it helps to clear up the user’s problem of how to specify the restaurant type. The wizard’s communicative competence is shown in the way she manages to provide information about the application and not just from the application.3

In our examples, the wizard’s cooperativeness is not restricted to pointing out and correcting false presuppositions, but also includes active initiatives concerning compensatory information and re-routing. This resembles overanswering in [18]: additional information, assumed to be useful or relevant for the user is included in the response. As pointed out by [13], such additional information is already available as a side-effect of the search which the system needs to perform to give an answer to the direct question in the first place.

The inclusion of compensatory information in the response depends on the type of failure and on the importance of this information in the communicative situation. In (2), for instance, the user asks follow-up questions after having received a list of restaurants, while the wizard has no information in the database concerning the requested information. In (2a), the wizard’s world knowledge contains general information about the opening hours of restaurants, and this is included in the response; in (2b), the request has been associated with information in the world model which is not general but a special feature of each individual restaurant, and thus a rerouting is offered. Finally, in (2c), no relevant information can be found, and a statement on the limits of the knowledge must be given.

\[ \begin{align*}
(2) & \quad (a) \\
& \quad \text{User2: which ones are open after midnight} \\
& \quad \text{Wizard2: The YP cannot be specific; however,} \\
& \quad \text{in general such restaurants are open until 2 am }
\end{align*} \]

\[ \begin{align*}
& \quad (b) \\
& \quad \text{User3: do any of them have a special student night} \\
& \quad \text{Wizard3: The YP cannot provide this information; please contact the restaurants concerned}
\end{align*} \]

\[ \begin{align*}
& \quad (c) \\
& \quad \text{User1: are they all licensed to sell alcohol} \\
& \quad \text{Wizard1: The YP cannot provide this information}
\end{align*} \]

Another kind of compensation is exemplified in dialogue (3) where the wizard takes the initiative and requires the user to explicitly accept or reject the compensation. The wizard recognizes metonymic relation between ‘food’ and ‘restaurant’ (if the user wants to know about American food, this could mean that she is looking for an American restaurant), and so the user’s request is re-interpreted with respect to the task that the wizard is able to perform, namely to give YP-information about restaurants.

\[ \begin{align*}
(3) & \quad \text{User1: describe what american food is like} \\
& \quad \text{Wizard1: Sorry I do not know. Would you like details of the American restaurants?}
\end{align*} \]

2.2 KNOWLEDGE LIMITS

When the limits of the speaker’s knowledge are encountered, it is commonly assumed that the cooperative reaction should include some explanation on the failure instead of a simple no [14].

3The adjacency pair Wizard2/User3 is also an interesting example of non-conventional encoding of communicative acts. The acts ‘ask’ and ‘inform’ are expressed in contributions whose literal force is exactly the opposite: the wizard asks the user about the restaurant type by providing a list of alternatives in a declarative statement, whereas the user selects one by an (elliptical) question.
In the examples (2)-(3) the wizard acts as a helpful information provider and always selects the strategy whereby mutual knowledge is established by giving all relevant information to the user. However, the sufficient amount of mutual knowledge can vary from general similarities to rather detailed local information. Moreover, an appropriate way to react does not require truthfulness in the sense that the response details one's knowledge limits. For instance, in (2a), it is not necessary to tell explicitly whether 'after midnight' is truly understood or if the response is simply based on the general information about the concept 'opening hours'. The response strongly implies that the helpful information is provided because the meaning of the whole contribution is understood, but even if this were not the case, the user could be satisfied with the inference which she can draw from the response “in general such restaurants are open after midnight”, most probably without even noticing that 'after midnight' is not understood.

In this example, it is rather safe not to bother the user with the problematic 'after midnight', but this need not always be the case. If the user continues with Oh, sorry, I meant which will be open until midnight, there is a high risk of misunderstanding and a need to recover. The user may think that the lack of information is due to her wrong specification, but the wizard is unable to locate the source of misunderstanding, since the user has repeated almost the same request. Also, 'open until 2am' implies 'open after midnight', but it does not imply 'open until midnight', and thus, while the first inference is valid, the inference “in general such restaurants are open until midnight” is not. Ignoring the subtle difference in the user questions, false inferences can be drawn and misunderstandings caused later in the dialogue.

On the other hand, if the speakers always play safe and make sure that all the references are understood so as not to give rise to false inferences, they are bound to hit their knowledge limits and spend a lot of time either in conveying to the partner information which is not relevant in the dialogue (I don't understand what you mean by 'after midnight'), or in asking the partner for nitty-gritty details (What do you mean by 'after midnight'?). Although in this way there is a relatively low risk of missing the communicative goal, the fluency of communication suffers. In [6], it is further argued that human agents often behave as if they have failed to plan an adequate response, since this strategy may turn out to be more efficient in the long run. Resource-bounded agents quickly reach their limits if they try to avoid all possible confusions, while recovery from vagueness and misunderstandings is relatively inexpensive in such a flexible mode as dialogue; thus it’s more rational to "plan to fail".

2.3 Conflicts

Since dialogue participants are capable of taking initiatives and selecting communicative strategies, they are also committed to their goals. Commitment to a goal means that the agent will give up the goal only if achieved, if believed to be impossible to achieve, or if the reasons for the goal are no longer true [8]. Thus the agents may find themselves in situations where they want to pursue conflicting goals with the partner.

In information-seeking dialogues, potential conflict situations occur if the speaker initiated a goal but the partner fails to respond to it. The speaker must then prioritize the goals and coordinate communication in such a way that her goals (considered as of the top priority) can be pursued further while the partner's goals can be duly attended to as well. To resolve conflicts, some measurement of the agent's preferences is thus required. In information-seeking dialogues, the preferences seem to be implicitly encoded in contextual facts: the degree of thematic relatedness between the partner's contribution and the speaker's previous contribution provides an estimate about how closely the partner wants to continue the current goal, while the fact who initiated the goal gives an estimate about the speaker's commitment in achieving the goal.

In dialogue (4), the question about the location is not answered, but the user's contribution concerns the type of restaurants. The situation is commonly referred to as 'subdialogue', but as this term is associated with a dialogue structure, which is not an appropriate way to describe dialogues [12], the situation is regarded as a conflict: the system needs to decide whether to continue with the user's goal (about restaurant types) or with its own goal (about the location).

(4) U1: I'm looking for a restaurant.
S1: In which area?
U2: What types of restaurants do you list?
S2: Indian, Mexican, Chinese, Italian, Thai.
U3: Try Indian in Rusholme.
S3: Please wait....
   Here's the list: <>

The user is allowed to 'manage' the dialogue, i.e. the system retains its own goal for later processing, and adopts the user's goal instead. The rationale behind this is that because the user question is thematically related to the whole dialogue, the user
may have a more general plan which she is executing but which the system is unaware of; by letting the user pursue her own goal, this plan will be gradually revealed and as a side-effect the system's current goal may become redundant. In example (4), the user indeed gives the missing information about the location together with the restaurant's type, and thus satisfies the postponed goal. It may also be that the user's goal is misunderstood, so following the wrong interpretation, the dialogue only gets into deeper misunderstandings. Moreover, it is always possible to return to an unfulfilled goal later, given that it is still relevant.

In example (5), however, the user has reacted with an unrelated question in a non-expected way, and it is not clear how this question is related to the current dialogue goals (looking for information on restaurants and specifying their location). The user question can be vague, too, i.e. *bus time tables* need not be known, but this does not affect the reasoning about a communicative strategy (although it can affect realisation of the chosen strategy on other planning levels): the user's commitment to the current dialogue goal is measured by the degree of her contribution's thematic relatedness to the previous topic, not by the reasons which the partner finds for this unrelatedness.\(^4\)

(5) U1: I'm looking for a restaurant
   S1: In which area?
   U2: Do you know bus time tables?
   S2: I thought we were talking about restaurants and wanted to know where you are looking for a restaurant.

Since the speaker's own communicative goals are prioritized over the partner's sudden topic changes, the strategy is to remind the user of the topic of the dialogue as well as the question she is expected to answer. By formulating the contribution as a statement, the initiative is given to the user, whose right to manage the dialogue by initiating new topics and abandoning her previously initiated but unfinished goals is thus acknowledged. However, being a cooperative, rational agent, the user is also expected to provide an explanation or a clarification related to the topic and change in mutual knowledge.

The above examples show how the wizard is able to make strategic decisions and clear up vagueness, knowledge limits and conflicts by providing helpful information, ignoring misunderstandings, and deciding on the goal priorities. The next section outlines how these decisions can be accounted for by the agent's rationality and cooperation.

3 PLANNING COMMUNICATIVE CONTRIBUTIONS

We have two basic assumptions about communicative situations: the dialogues are *collaborative activities* and the participants are *rational agents*. The former means that we need to model the agents' cooperation in their attempts to fulfill the goal, while the latter accounts for the agents' efforts in establishing enough mutual knowledge for the successful task accomplishment. Rationality is measured by the act's *operational appropriateness* (the act fulfills, or is a step towards the fulfillment of the goal), and *ethical adequacy* (the act does not cause obstacles to other agents), which pull in different directions. An act can thus be rational to different degrees depending on how well the agent thinks it fits goal fulfillment and ethics, but its rationality is always relative to what the agent believes in a given situation, see [11].

3.1 IDEAL COOPERATION

Rational agents are engaged in Ideal Cooperation [2], if they:

1. voluntarily strive to achieve the same purposes,
2. cognitively and ethically consider each other in trying to achieve these purposes
3. trust each other to act according to (1) and (2) unless they give each other explicit notice that they are not.

Ideal Cooperation describes the ideal standard but does not require the agents' full cooperation. As [9] points out, if agents are always in agreement and ready to adopt the other's goals, they are benevolent rather than cooperative. Ideal Cooperation captures the agents' basic attitude: the agents are willing to receive, evaluate and react to the partner's actions. In communication, the agents have at least one common purpose: transfer of information, and if this disappears there will be no collaboration nor communication.

In CDM, the first requirement of Ideal Cooperation is encoded in the reasoning about the joint purpose, while the justification of particular beliefs, desires and actions is based on the interaction of cognitive and ethical considerations, and implemented in communicative obligations. The trust is modelled by the bidirectional use of the principles in both analysis and planning.

3.2 JOINT PURPOSE

The communicative strategy, or *joint purpose*, is chosen on the basis of four contextual factors:
<table>
<thead>
<tr>
<th>response</th>
<th>central concept</th>
<th>goals</th>
<th>speaker initiative</th>
<th>partner initiative</th>
</tr>
</thead>
<tbody>
<tr>
<td>expected</td>
<td>related</td>
<td>unfulfilled</td>
<td>backto</td>
<td>follow-up-old</td>
</tr>
<tr>
<td></td>
<td>fulfilled</td>
<td>finish/start</td>
<td>follow-up-new</td>
<td></td>
</tr>
<tr>
<td>unrelated</td>
<td>unfulfilled</td>
<td>repeat-new, X</td>
<td>new question</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fulfilled</td>
<td>specify</td>
<td>new request</td>
<td></td>
</tr>
<tr>
<td>non-expected</td>
<td>related</td>
<td>unfulfilled</td>
<td>subquestion, X</td>
<td>continue</td>
</tr>
<tr>
<td></td>
<td>fulfilled</td>
<td>new dialogue</td>
<td>somethingelse</td>
<td></td>
</tr>
<tr>
<td>unrelated</td>
<td>unfulfilled</td>
<td>object, X</td>
<td>notrelated</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fulfilled</td>
<td>specify-new</td>
<td>new st-request</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Possible communicative strategies if the contextual factors are assigned binary values. Those marked with X refer to conflict situations, where the speaker must decide whether to persist with her own goal or take up the partner's goal.

1. Partner related:

   (a) **Expectations**: whether or not the partner's contribution conforms to the expectations evoked by the speaker's previous contribution,

   (b) **Central Concept (CC)**: whether or not the partner's contribution is thematically related to the previous CC,

2. Speaker related:

   (a) **Initiatives**: whether or not the speaker has the initiative,

   (b) **Goals**: whether or not the speaker has unfulfilled goals.

Each aspect is treated as a call to the Context Model, and the combination of the values determines the joint purpose. For instance, if the partner's response was expected and related to the current CC, and the speaker has unfulfilled goals and the initiative, then the speaker can go 'back to', and adopt one of the pending goals that has not become fulfilled or irrelevant in the course of the dialogue. In the current implementation, the aspects are assigned binary values, and we have $2^4 = 16$ joint purposes, shown in Figure 1.

3.3 **COMMUNICATIVE OBLIGATIONS**

Communicative obligations are regarded as norms that rational, cooperative agents follow, or, from the analysis point of view, as assumptions about the partner's rationality and cooperativeness. They concern the speaker's sincerity (truthfulness with regard to the current intentions), motivation (justification of the current intentions) and consideration (consideration of the partner's intentions). Motivation checks that the context warrants the epistemic facts included in the goal and may require some of the facts to be dropped, whereas consideration checks consistency of the facts with respect to the context, and may require some facts to be included in the goal. Figure 2 lists communicative obligations in their verbal form.

**Sincerity**: do I know/have evidence for this?

1. Everything that the speaker asserts or implies is true unless otherwise explicitly stated.

**Motivation**: can I say this?

1. Everything that the speaker wants to know or wants the partner to do is motivated except if the speaker cannot take the initiative on it.
2. Everything that addresses what the partner wanted to know or wanted the speaker to do is motivated, except if the speaker cannot disclose the information or do the act.
3. Everything that is related to Central Concept is motivated if not already known.
4. Everything that informs of inconsistency is motivated if not already known.

**Consideration**: may I say this?

1. If the partner's goal cannot be fulfilled (presuppositions are false, facts contradictory, no information exists), it is considerate to inform why (give explanation, add compensation, initiate repair).
2. If the amount of information is too much (according to some criterion that constrains the amount of new information that can reasonably be included in a contribution), it is considerate to split the information into manageable portions.
3. If the response repeats previous information, it is considerate to paraphrase or leave this implicit unless the information is assigned a special emphasis.
4. If the partner's response is irrelevant, it is considerate to inform of the irrelevance, given that the agent has unfulfilled goals.
5. If the partner's response is unrelated and the partner has the initiative, it is considerate to ensure that the partner wants to change the topic, given that the agent has unfulfilled goals.
6. If the partner did not request a piece of related information, it is considerate to include this explicitly in the response, given that the agent intends to close the topic.

Figure 2: Communicative obligations.
4 The CDM System

In the Constructive Dialogue Model (CDM), the participants are rational agents who follow the principle of Ideal Cooperation [12]. The agents have two kinds of goals: task goals (t-goals) which concern the underlying task that they try to accomplish, and communicative goals (c-goals) which deal with providing and eliciting information. Communication starts when the agent needs to obtain information to accomplish a task goal. The agent and her partner become engaged in constructing a model of the 'joint purpose', which activity continues until both agree that the task is accomplished. The dialogue consists of a sequence of contributions ([3]), and has a flat structure, created incrementally so that the contributions are locally coherent. Contributions contain context changing potential ([5]), and they are formed as a reaction to the updated dialogue context.

4.1 System Architecture

The main reasoning component is the Dialogue Manager (DM) which has access to three static knowledge bases: Communicative Principles (knowledge about rational cooperative communication), Application Model (knowledge about tasks and roles in regard to the application backend; in our case: to provide information from a small database of car hire companies and restaurants) and a World Model (general knowledge about the entities and their relations in the world).[^3] The key resource of the DM is the Context Model which records the goals, intentions, beliefs, discourse referents, Central Concept, NewInfo etc. of each contribution. The Context Model is queried and dynamically updated during the dialogue session.

In this paper we concentrate on the system's reaction, the detailed task division of which is given in Figure 3. The planner differs from hierarchical planners in that it operates on dialogue states and reacts to the changes in the Context Model. The reaction is determined in two steps: first, the user's c-goal is evaluated, finishing with the system's c-goal; second, the response is specified up to the semantic representation of the c-goal.

The evaluation of the user's c-goal starts with checking that the basic requirements are fulfilled. These requirements refer to contact (there exists a communication channel between the participants) and perception (the contributions are recognised as contributions). In the implementation, the former is fulfilled by the fact that the user is connected to the system in the first place, while the latter deals with the evaluation of the reliability (parse result, analysis) of the user goal.[^6] Thus the system can launch a repair strategy, rather than a normal reaction, to a c-goal which is unreliable.

---

[^3]: Linguistic knowledge is encoded in a linguistic lexic and grammar, and not discussed here.

[^6]: In spoken dialogues this step is important in regard to the reliability of different results of speech recognition.
coherence and propositional content (higher level linguistic requirements like sentence structure and referring expressions). The result is a semantic representation which is the input to a surface generator and will be realised as a string of words.

4.2 REPRESENTATIONS

The prototype system is implemented in Prolog and the contextual facts are represented as Prolog-type facts. The World Model uses neo-Davidsonian event representation (the predicates have an extra argument referring to the event itself), and the Application Model provides mappings from World Model concepts to task and role related facts.

Beliefs and intentions are expressed in the attitude language which contains two operators know and want, referring to believing and intention, respectively. The operators are two-place predicates whose first argument refers to the agent holding the attitude, and the second argument to the content of the attitude. The content of know is a conjunction of world model concepts, and the content of want is a belief, an intention, or a particular predicate do which refers to performing an act.⁷

An epistemic fact is a belief that describes the agent’s mental state. A communicative goal is a set of epistemic facts that describes the desired mental state. For instance, the epistemic fact

\[
\text{know}(s,[\text{wantEvent}(u,w,l), \text{listOf}(1,r,L), \\
\quad \text{user}(u), \text{restaurant}(r)])
\]

describes a state where the system s knows that there exists a wanting event w such that the user u wants a list l of restaurants r (to be instantiated in L).⁸

If formulated as part of a c-goal, it refers to a desired future state where s knows if (it is true that) u wants a list of restaurants.

A communicative intention is an intention to act so that the goal will be true. The above c-goal gives rise to the intention: s intends that s knows if (it is true that) u wants a list of restaurants.

\[
\text{want}(s, \text{know}(s,[\text{wantEvent}(u,w,l), \text{listOf}(1,r,L), \\
\quad \text{user}(u), \text{restaurant}(r)]))
\]

A c-goal may contain several facts, but give rise only to a (sub)set of consistent intentions.

An expressive attitude is a communicative intention carried by a contribution. An evocative attitude is an intention intended to be evoked in the partner by uttering the contribution. Their contents differ if the propositional content of the contribution can be further interpreted with respect to contextual knowledge, in which case the evocative attitudes contain the inferred content. They cannot be associated with the speech acts' preconditions and effects, since they encode attitudinal updates of the context rather than definitions of possible communicative acts (see [12]).

Communicative obligations are inference rules over Context Model, of the form:

\[
\text{motivation}(\text{N}, \text{Attitude if} \\
\quad [\text{CxFact1}, \text{CxFact2}, \ldots, \text{CxFactN}]).
\]

\[
\text{consideration}(\text{N}, \text{Action if} \\
\quad [\text{Att1}, \text{Att2}, \ldots, \text{AttN}]).
\]

N is the index of the rule and CxFact is a predicate describing Context Model. Attitude is an intention want(Speaker, know(Partner, F)) or want(Speaker, know(Speaker, F)) that is warranted if the contextual facts hold, while Action is a procedure to be undertaken if the attitudes hold. If the list of contextual facts or attitudes is empty, Attitude or Action is always true.

4.3 EXAMPLE

Consider again Example (1). The analysis of the first user contribution User1 is given in Figure 4.

Assuming that the basic requirements are fulfilled, the system starts the evaluation of the user’s c-goal. The joint purpose is determined on the basis of the contextual facts: the contribution is expected (a request to give information about restaurants), it is unrelated to a previous topic (trivially, since no previous topic exists), the partner has the initiative, and the system has no unful-
filled c-goals. This leads to the new-request strategy, and to the adoption of the user's evocative attitude as the system's attitude ("system intend that user know that there are good restaurants"). This attitude (or actually its content) is filtered with respect to the Application Model, resulting in the system's c-goal "user know that there exists 48 restaurants, r1,...,r48":

\[
\text{know}(u, \text{exists}(1, \text{list0f}(1,r,[r1,\ldots,r48]), \text{cardinality}(1,48), \text{restaurant}(r)))
\]

This goal is further specified by the communicative obligations. Since the cardinality is considered too big, Consideration(2) directs the system to split the number of restaurants into manageable portions. In this case, cardinality is restricted by the type and location parameters, so the system wants to know their values. The user specification type(t,r,good) appears irrelevant, and according to Consideration(4), the user should know about this. Thus the system's c-goal becomes a set of facts which describes the desired future state as "user knows there is no such type as 'good', system knows the location of the restaurants, system knows the type of the restaurants":

\[
\text{know}(u, \text{not} (\text{type}(t,r,\text{good}), \text{restaurant}(r)))
\]
\[
\text{know}(s, \text{location}(p,r,\text{c-o-m}), \text{restaurant}(r))
\]
\[
\text{know}(s, \text{type}(t,r,\text{T}), \text{restaurant}(r))
\]

At the beginning of the dialogue, the speaker usually makes no assumptions about the partner's knowledge or the mutual knowledge, and thus all the information related to the central concept 'restaurant' is motivated (Motivation(3)), and given to the user. If the speaker knows that the specifications are known to the partner (e.g. they have been mentioned earlier in the dialogue), the two latter facts can be omitted. If the speaker takes a risk of not informing the partner of the shortcomings of her request, the first fact need not be realised in the surface contribution.

The analysis of the contribution User2 looks as in Figure 5 (assuming that the relation between 'restaurant' and 'food' is known, and the misspelt 'somw' is considered insignificant in regard to the reliability of the goal). Now the joint purpose becomes backdo: the response is expected and thematically related, the system has unfulfilled goals and the initiative. Thus system 'goes back' to the previous unfulfilled goal and tries to satisfy it in the updated context. The system intention \text{want}(s, \text{know}(s, \text{type}(t,r,\text{T}), \text{restaurant}(r)))) is not fulfilled, so the system attempts to fulfill this intention again. As the intention is not assigned a special status, Consider-

Figure 5: Context Model after the user contribution

\text{spicy food in the centre of manchester. p, t and r represent instantiated concepts, c-o-m the location Centre of Manchester, and spicy is the type value 'spicy'.}

\text{NEW INFO:}
\text{location(p,r,\text{c-o-m}), type(t,r,\text{spicy}), restaurant(r)}
\text{USER C-GOAL: know(s,}
\text{\text{location}(p,r,\text{c-o-m}), type(t,r,\text{spicy}), restaurant(r)))
\text{CENTRAL CONCEPT: restaurant(r)}
\text{EXPRESSION ATITUDES:}
\text{intention: user intend that system know P:}
\text{want(u,know(s,}
\text{\text{location}(p,r,\text{c-o-m}), type(t,r,\text{spicy}), restaurant(r)))}
\text{assumptions: user know that system not know P:}
\text{know(u,not(know(s,}
\text{\text{location}(p,r,\text{c-o-m}), type(t,r,\text{spicy}), restaurant(r))))}
\text{EVOCA TIVE ATITUDES:}
\text{intention: user intend that system intend that}
\text{system know P:}
\text{want(u, want(s,know(s,}
\text{\text{location}(p,r,\text{c-o-m}), type(t,\text{spicy}), restaurant(r))))}

5 Conclusion

This paper has discussed how the general principles of rational and cooperative communication can be used in generating communicatively adequate responses in information seeking dialogues. The argumentation has concentrated especially on the communicative strategies by which the participants can establish sufficient mutual knowledge for fulfillment of the task, and it is claimed that the ability to clear up vague or partly understood contributions is also one of the distinctive features in improving dialogue systems' robustness.

Constructive Dialogue Management (CDM) is presented as a new approach to dialogue modelling. It advances from previous research in that communicative principles are explicitly taken into consideration, and it provides a uniform analysis of various phenomena that have been separately studied in previous research: determination of an appropriate communicative goal and realisation of this goal in a contextually relevant way. Communicative principles are inference rules based on the agents’ rationality. They gradually specify the goal and function on the following levels:

1. determination of a joint purpose on the basis of expectations, initiative, topic, and unfulfilled c-goals.
2. filtering of the joint purpose with respect to
domain knowledge.
3. specification of the goal with respect to the
communicative obligations.

Future work will concern the effect of linguistic in-
formation in the gradual specification of a system
goal, and the interaction between communicative
principles and linguistic phenomena. The gener-
ality of the system is currently being improved by
applying the basic framework to other types of di-
alogues (e.g. task-oriented dialogues) and other as-
pects of communication (e.g. reasoning about the
system’s role, the task it is involved in, and the
expertise level of the user). These extensions enable
us to study the validity of communicative prin-
ciples in a wider range of realisation contexts and
also to augment the rules to cover subtler differ-
ences between realisation alternatives.

REFERENCES

[1] J. F. Allen. Recognizing intentions from nat-
ural language utterances. In M. Brady and
B. Berwick, editors, Computational Models
1983.

tion and Cooperation. Department of Linguis-
tics, University of Göteborg, 1976. Göten-
burg Monographs in Linguistics 2.

activity analysis of a Wizard of Oz experi-
ment. Technical Report PLUS working pa-
er, Department of Linguistics, University of

[4] W. J. Black, H. C. Bunt, F. Dols, C. Donzella,
G. Ferrari, R. Huidan, W. G. Imlah, K. Joki-
nen, T. Lager, J.-M. Lancel, J. Nivre,
G. Sabah, and T. Wachtel. A pragmatics-
based language understanding system. Espir-

[5] H. C. Bunt. Dynamic interpretation and di-
alogue theory. In The Second Venice Work-
shop on Multimodal Dialogue. Acquafrredda di

[6] J. Carletta. Planning to fail, not fail-
ing to plan: Risk-taking and recovery in
task-oriented dialogue. In Proceedings of

as a collaborative process. In P. R. Cohen,
J. Morgan, and M. E. Pollack, editors, In-

tentions In Communication, pages 483–493.

interaction as the basis for communication. In
P. R. Cohen, J. Morgan, and M. E. Pollack,
editors, Intentions in Communication, pages

computer models of cooperative dialogue, ac-
knowledging multi-agent conflict. Technical
Report 172, University of Cambridge, Com-
puter Laboratory, 1989.

proving the Efficiency of Human-Machine Di-
ologue. A Computational Model of Variable
Initiative and Negotiation in Collaborative
Problem-Solving. PhD thesis, Duke Univer-

editor, Rational Agency: Concepts, Theories,
Models, and Applications, pages 89–93. Pro-
cedings of The AAAI-95 Fall Symposium,
MIT, Boston, 1995.

[12] K. Jokinen. Goal formulation based on com-
unicative principles. In Proceedings of the
16th COLING, 1996.

[13] A. Joshi, B. L. Webber, and R. M.
Weischedel. Preventing false inferences. In
Proceedings of the 10th COLING, pages 130–

a portable natural language system. In M.
Brady and B. Berwick, editors, Computational

[15] K. McCoy. Correcting object-related miscon-
ceptions: How should the system respond.
In Proceedings of the 10th COLING-84, Stan-

Dialogues. Interpreting and Responding to
Questions in Context. MIT Press, Cam-

[17] J. Nivre (Ed.). Corpus collection and analy-


[18] W. Wahlster, H. Marburger, A. Jameson, and
S. Buseman. Over-answering yes-no ques-
tions: Extended responses in a NL interface
to a vision system. In Proceedings of the 8th
Dialog Grounding for Speech Recognition Systems

Elizabeth A. Hinkelman
Kurzweil Applied Intelligence
411 Waverly Oaks Road
Waltham, MA 02154 USA
elizh@kurz-ai.com

ABSTRACT

We apply existing dialogue grounding models to an experimental speech recognition system. Due to the state of the art in speech recognition, this leads to what is essentially a second formulation of the model. We then outline a study in progress using this model, discuss a representative dialogue segment in terms of the model, and use it to point to areas in which the system performance could be improved.

1 BACKGROUND

We consider an experimental large-vocabulary speech recognition system designed to handle both isolated word dictation and word processing commands in natural language. The system presents the user with some graphic interface elements specific to word processing, and some specific to conducting dialogue. Its production of natural language responses is limited to predefined strings, however, this limitation impinges on the present analysis only slightly.

We are interested in characterizing the dialogue behavior of such a system, evaluating it, and improving on it.

2 THE NATURAL LANGUAGE DIALOGUE GROUNDING MODEL

We take as our starting point the model of dialogue grounding presented in Traun Hinkelmann 1992\(^1\). In this model, dialogue is regarded as having four levels: turn-taking, grounding, core speech acts, and argumentation acts. We focus on the special role of grounding, the shared process of establishing a core speech act through talking. The grounding process can be modelled by a finite state automaton whose transition labels are

- `init`
- `continue`
- `ack`
- `req-ack`
- `repair`
- `req-repair`
- `cancel`

It can be seen that the job of the application program is to carry out core speech acts that arise from this process. The job of the speech recognition system, then, is precisely to perform turn-taking and grounding.

We take this model of human-human interaction as a guide to desirable properties of human-machine interactions, but we cannot expect an exact mapping. This is for two reasons. First, computers have different strengths and weaknesses than people, and good design decisions take advantage of the strengths and compensate for the weaknesses. Second, the state of the art is simply not that advanced.

3 DIALOGUE GROUNDING FOR SPEECH RECOGNITION SYSTEMS

Consider user interactions with a bare speech recognition engine. Basically, the user says something and the system produces an interpretation either correct or erroneous. On the basis of this

recognition and the implied core speech act (either dictation or command in the experimental system\footnote{we will not discuss the details of the discrimination process here}), the system goes on to perform the associated action (insertion of an isolated word of dictation into a document, or execution of a command, respectively). In the human-human interaction model, the user's recognition of this associated action carries an implicit acknowledgement by the hearer that the original utterance was understood. In human-machine interaction the same is true, except that the machine is often in error and hence often performs the wrong action. This effectively signals a failure of understanding.

The experimental system under discussion always provides a text display of the utterance it recognized, in addition to performing the associated action. Again bearing in mind the likelihood that this is in error, we will refer to the text display as an echo action, whether it is in error or not. This new kind of grounding action combines the role of an acknowledgement in the human-human system with the role of a request for repair. In the actual experimental system, the role of request for repair is enhanced by the fact that the four most likely alternate hypotheses are provided as part of the text display as well.

At this point our automaton has a start state (S0), a post-utterance state(S1), and a post-echo state(F) which is a final state. We just referred to the utterance as an init, but in fact the system would treat a second or subsequent attempt in the same way as a true init. Therefore we'll refer to such an utterance as a try, and distinguish tries from retries only from a user perspective.

The experimental system's user interface supports three different ways of performing a repair. In order from lightest to heaviest weight, they include

- requesting a listed alternate ralter,
- requesting a correction rcorr,
- requesting a recovery rreco.

\textbf{Ralter} (F \rightarrow S2) allows substitution of an alternate hypothesis with only a single grounding-level command and its corresponding display update (alter), (S2 \rightarrow F).

\textbf{Rcorr} (F-S3) allows the user to supply a replacement using keystrokes or voice-enabled spelling, a process involving multiple inputs and other allowed supporting events designed to reduce the number of inputs necessary; we won't delve into the details here. Rather, we ask that you remember that S3 is really an entire subautomaton, and two transitions are required to get out of it: a user acknowledgement of completing the correction (acorr), (S3 \rightarrow S4), and the actual correction insertion screen update (icorr), (S4 \rightarrow F).

Requesting a recovery (F \rightarrow S5) we rate as the heaviest because it means explicitly undoing the previous core speech act before starting over (S5 \rightarrow S0) with the next try. Although it's potentially many events fewer than a correction, the cognitive load of taking two steps back in order to move forward is so strong as to outweigh basic corrections.

Each of these repair mechanisms adds transitions to our state machine. Ralter is a transition to a new state, with alter returning from that state to the final one. Rcorr is a transition into a new subautomaton, requiring a final user acknowledgement of the correction before the system's correction display update and return to the final state. Rreco is also a transition into a new state, with the system's recovery display update and a return to the start state.

There are two cases we haven't covered. The first is when, after any user input (unrecog), the system totally fails to find an interpretation for it (nack). We thus have a move to a new state, followed by a system negative acknowledgement status message and return to the original state. The other case is when the system hears something that isn't speech addressed to it—other speaker, other addressee, other background noise, mouth noise. We refer to this simply as \textit{other}; there's a transition to a new state, and a noise system status message transition back to the original state. These are relatively rare in our illustrative dialogue, and we won't list the additional states and transitions in detail.

Thus the overall list of actions is

- try (user)
- echo (system; generates a core speech act and a responding action)
- ralter (user)
- alter (system)
- rcorr (user; leads to a subautomaton)
- acorr (user)
• icorr (system)
• rrreco (user)
• reco (system)
• unrecog (user)
• nack (system)
• other (user or background)
• noise (system)

4 A Usability Study

An industrial software usability study is essentially a watered-down psychology experiment: a test population of users is asked to perform a task appropriate to the software being studied, and observations are made. In the case of a natural language processing system, this yields a set of human-machine dialogues.

We have performed such a usability study on our experimental system; the task was a word processing task including some dictation but focussed mainly on commands. The users were chosen to be familiar with the word processor used, and unfamiliar with speech recognition. Eight sessions with novice users were recorded. The users received a twenty-minute online tutorial on speaking to the system, and viewed brief recordings of successful and less successful sessions on commands for word processing. This covered all grounding functionality indirectly, with the exception of the more complex aspects of correction.

Overall numerical results from this study are not yet available. For this paper, we will use a representative hundred-utterance (two hundred turn) segment to illustrate general methodology and trends.

Each machine-endpointed utterance was transcribed by a professional transcriptionist; the results were paired with the actual recognition results and both were labelled according to our human-machine interaction model. The two labels resulting were compared with each other to determine the success or failure of the individual grounding acts.

4.1 Breakdown, Recognition, and Initiating Events

Of the hundred recognition events, sixty-nine were successfully handled by the system. The breakdown is

• 48 try, 29 successfully handled
• 27 rrreco, 27 successfully handled
• 22 other, 10 successfully handled
• 3 ralter, 3 successfully handled

Of the 48 tries, 36 were first tries. 24 of these were successfully handled. Of the 12 retries, 5 were successfully handled. This means that the user saw a rate of 24 initial successes out of 100, for an overall initial success rate of one in four, or an average of four utterances (eight counting system responses) per core speech act. The distribution of grounding acts per core act was 25, 9, 9, 6, 5, 4 (plus three occurring after the end of our hundred), 4, 4, 3, 2, 2, 2, and 25 ones (all x 2 to include system responses).

It could be argued that we should subtract out the twenty or so other utterances occurring in the longest core act, on the grounds that they are due to a discussion between the user and the experimentors. We didn’t for the first analysis, because the user needed this help to continue the task and because it’s likely enough that the user would forget to turn the microphone off at such a time. If we do subtract them out, the numbers become 24 initial successes out of 80, for an overall success rate closer to one in three, or an average of 3 1/3 utterances (seven counting system responses) per core speech act.

This segment is taken from near the beginning of a session, from an average rather than a relatively successful user. Some sessions have much better success rates, as would any experienced user including the same user later in the session.

4.2 Repair

Of the three types of repair, this user mainly requested recoveries. This was due to the large amount of conversation background, with its tendency to lead to spurious insertions, to a tendency by recognition components to prefer dictation interpretations, and to the user’s not having enough attention available to consider the alternative recognition choices list in some cases. This user did occasionally request listed alternates, and didn’t in this segment use the more circuitous correction mechanism at all. (note: we need to check whether the user ever inserted keystrokes directly into the document before making final tabulations). Note that overall recognition for these events is perfect. This is due to engineering
these acts to make them more reliable, and to the fact that users were explicitly trained to use them during the tutorial phase.

Perfect repair recognition doesn't mean that repair is easy, though. Consider the following segment, in which the user is trying to insert a single ampersand character into the document. The first line of each pair is the transcription of the user's act with label, and the second, the system response:

```
init d plus
echo d live
rreco delete that
reco delete-that
retry and
echo d and
rreco delete that
reco delete-that
retry plus
echo d plot
other background-noise
echo d noon
rreco delete that
reco delete-that
other background-noise
noise Bead a Noise
rreco delete that
reco delete-that
```

Backing out of a misrecognition really needs to be made much easier.

4.3 Other Improvements

There are several other areas for improvement that can be identified on the basis of this analysis. For example, if we take the other category seriously, we can see that humans use additional information sources to determine whether they are intended addressees of utterances, and attempt to use these sources to determine when to ignore acoustic input. That would give better system success for the other category.

A second issue is continuation. Since transcriptionists were given machine-identified endpoints, they were not in a position to identify commands that were split into two events (generally interpreted as dictation by the system). Thus we're not in a position to identify the frequency of misrecognitions due to hesitations or pauses within commands. The system could use better treatment of pauses.

A third issue that falls out of the dialogue model is that there isn't really a way to cancel, that is, to back out of a speech act in the middle. This would arise during correction, or during occasional application events such as repositioning a window which don't appear atomic to the user.

5 Conclusion

We have been able to recast the grounding model of human-human communication to describe the behavior of current speech recognition systems, and to use this model to identify areas for improvement.

6 Appendix

The full two hundred turns appears below:

```
init d April
echo d April
init d R
echo d are
alter take two
alter take-two
init d plus
echo d live
rreco delete that
reco delete-that
retry and
echo d and
rreco delete that
reco delete-that
retry plus
```
echo d plot

other background-noise

echo d noon

rreco delete that
reco delete-that

other background-noise
noise Heard a Noise

rreco delete that
reco delete-that

init d D

echo d the

rreco delete that
reco delete-that

retry D

echo d the

rreco delete that
reco delete-that

retry D

echo d being

rreco delete that
reco delete-that

init d status

echo d adding

rreco delete that
reco delete-that

retry d status

echo d contain

rreco delete that
reco delete-that

retry d D

echo d be

ralter take two

alter take-two

init d status

echo d status

init d report

echo d report

init flash
echo d flash

other background-noise

echo d Britain

rreco delete that
reco delete-that

other background-noise
noise Heard a Noise

init d Marion
echo d Marion

init d Spivak
echo d effect

other background-noise
noise Heard a Noise

rreco delete that
reco delete-that

retry d Ruth

other background-noise
noise Heard a Noise

init d status
noise Heard a Noise

other background-noise
echo d who

retry d status
echo d a

other background-noise
echo d an

other background-noise
echo d who

retry d D
echo d be

other user-misspoke
noise Heard a Noise

init d status

other background-noise
echo who

init d report

other background-noise
noise Heard a Noise

echo d report
init c next line
echo c next-line

init c next line
echo c next-line

init c next line
echo c next-line

init c next line
echo c Wd> next line

init c next line
echo c Wd> next line

init c next line
echo c next-line

init c next line
echo c next-line

init c next line
echo c next-line

init c select word
echo c Wd> select word

init c underline word
echo c Wd> underline word

init c select word
echo c Wd> the ninth word

retry c select word
echo c Wd> select word

init c bold word
echo c Wd> bold the word

init c next word
echo c Wd> next word

init c under
echo d under

init c select
echo d time

other background-noise
noise Heard a Noise

rreco delete that
reco delete-that
Response Generation in Collaborative Dialogue Interactions

Jennifer Chu-Carroll*
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA
E-mail: jchu@cis.udel.edu

ABSTRACT

In collaborative planning dialogues, since the agents have different beliefs about the domain and about one another, it is inevitable that conflicts arise among them during the planning process. In order for a computer system to appear collaborative in such an environment, it must be capable of detecting the conflicts as soon as they arise and of resolving them in an efficient and effective manner. This paper describes an implemented system, CORE, which generates cooperative responses to user utterances during collaborative consultation dialogues. Given a user proposal, CORE is capable of evaluating the proposal based on its private knowledge to determine whether to accept or reject the proposal. In situations where a relevant conflict is detected, CORE is able to initiate collaborative negotiation for conflict resolution by first identifying the aspect of the rejected proposal which it will explicitly refute, and selecting appropriate justification to present to the user in refuting the identified beliefs. Furthermore, by capturing the collaborative process in a recursive Propose-Evaluate-Modify framework, CORE is able to handle embedded negotiation subdialogues.

1 INTRODUCTION

In collaborative planning dialogues, since the agents are often autonomous and heterogeneous, it is inevitable that conflicts arise between them during the planning process. In order for a response generation system to appear to be collaborative, it must be able to detect these conflicts as soon as they arise, and to engage in negotiation with the other agent(s) to resolve the relevant conflicts.

My analysis of naturally occurring collaborative planning dialogues shows that when an agent detects multiple related conflicts in another agent’s proposal, she often does not explicitly address each detected conflict. Instead, she will select a subset of these detected discrepancies in the agents’ beliefs to explicitly refute in order to resolve the conflict(s) relevant to the task at hand. Thus, a collaborative system must employ a strategy that allows it to select from among the detected conflicts those that it believes will most effectively resolve the relevant conflict(s) relevant to the agents’ current task. Furthermore, in order to successfully refute the selected beliefs, the system must also be able to select appropriate justification to support its view regarding these beliefs.

This paper presents a computational model for engaging in negotiation to resolve conflicts in the agents’ beliefs during collaborative planning. The system 1) detects conflicts in the agents’ beliefs and initiates a negotiation subdialogue only when the conflict is relevant to the task at hand, 2) identifies the most effective aspect of the rejected proposal to address in its pursuit of conflict resolution when multiple conflicts arise, and 3) selects appropriate evidence to justify the system’s beliefs during conflict resolution. In addition, the system captures the negotiation process in a recursive Propose-Evaluate-Modify cycle of actions, thus enabling it to handle embedded negotiation subdialogues.

2 MODELING COLLABORATION

In order for a computer system to participate in a planning process in a collaborative manner, it must be developed based on an underlying model for capturing collaborative activities. This model would then facilitate the recognition of agent intentions and the generation of cooperative responses. A number of researchers have proposed formal models for collaboration [12, 1, 11], and Sidner has developed an artificial language for modeling collaborative discourse [26, 27]. However, none of these models have been adopted as the basis for a response generation system.

Sidner’s artificial language captures collaborative
planning as a sequence of proposal/acceptance and proposal/rejection actions. In other words, she treats collaborative planning as a sequence of one agent making a proposal to another agent, and the second agent either accepting or rejecting the proposal. In cases where the second agent accepts the proposal, the proposal is considered a mutual belief between the two agents; on the other hand, if the second agent rejects the proposal, the proposal is deleted from the stack of open beliefs. It should be noted that in Sidner's model, a proposal is deleted from the stack of open beliefs as soon as an agent indicates rejection of the proposal, regardless of whether this rejection is accepted or rejected by the other agent. My analysis of naturally occurring collaborative planning dialogues [8, 28], on the other hand, shows that an agent does not always immediately accept another agent's rejection of her proposal. Instead, she will sometimes provide further evidence to support her proposal or provide evidence against the other agent's reasons for rejecting her proposal. In Sidner's model, the agent's further pursuit of her original proposal will be treated as her making a new proposal. In other words, her model fails to explicitly represent the relationship between the rejected proposal and the new proposal (which are in fact the same proposal), and thus is unable to capture negotiation processes.

I contend that in modeling collaborative planning dialogues, it is important that negotiation processes be explicitly represented. This explicit representation of negotiation not only will facilitate the recognition of user intentions during conflict resolution, but also will help guide the system's response generation process in producing cooperative responses. Based on my analysis of collaborative planning dialogues, I extended Sidner's proposal/acceptance and proposal/rejection sequences, and developed a framework that models collaboration in a Propose-Evaluate-Modify cycle of actions [4]. This model views collaborative planning as agent A proposing a set of actions and beliefs to be incorporated into the shared plan being developed, agent B evaluating the proposal based on his private knowledge to determine whether or not he accepts the proposal and, if not, agent B proposing a set of modifications to A's original proposal. These modifications will again be treated as a proposal and be evaluated by A, and if conflicts arise, she may propose modifications to B's previously proposed modifications, resulting in a recursive process.

To illustrate how the Propose-Evaluate-Modify cycle of actions captures collaborative planning dialogues, consider the following exchange taken from a transcript of naturally occurring course advisement dialogues [8]. In this dialogue, an advisor (A) is helping a student (S) plan his schedule:

1. S: I was going to say two [courses] this time and three next time.
2. A: And if you take two and then don't pass one, you also would be slightly behind.
4. But then if I take two, the probability is much higher that I'll do well in both of them.
5. Whereas if I take three...
6. A: Right.
7. People do take two, so...

In utterance (1), S proposes a plan of taking two courses this semester and three next semester. A evaluates this proposal based on her private beliefs and decides that taking three courses this semester and two next semester is a better alternative than S's proposal. Thus in utterance (2) A points out the disadvantage of S's proposal as a means of implicitly conveying her intention to modify S's proposal. S evaluates A's proposal of modification, decides that although A's reason for suggesting the alternative is a valid one (utterance (3)), his original proposal still constitutes a better plan; thus in utterances (4) and (5), S provides his evidence to support his original proposal of taking two courses this semester and three next semester as an attempt to modify A's belief that taking three courses this semester is a better alternative. A again evaluates S's new proposal, which consists of the beliefs conveyed by utterances (4) and (5), and in (6) and (7) accepts both S's new and original proposals. Thus, the Propose-Evaluate-Modify model successfully accounts for each agent's actions in initiating and responding to negotiation subdialogues during the collaborative planning dialogue above.

The rest of this paper describes an implemented plan-based response generation system, CORE, based on this Propose-Evaluate-Modify framework. Given a user proposal, CORE first evaluates the proposal based on its private knowledge to determine whether to accept or reject it. In situations where CORE detects a relevant conflict and rejects the proposal, it initiates a collaborative negotiation subdialogue to resolve the detected conflict in an attempt to modify the user's original proposal. The following sections describe the evaluation and modification processes, as well as the response generation process that ensues as part of these processes.
3 EVALUATING USER PROPOSALS

In order to capture the agents' intentions conveyed by their utterances, my model of collaborative negotiation utilizes an enhanced version of Lambert and Carberry's dialogue model [14] to represent the current status of the interaction. The enhanced dialogue model has four levels: the domain level which captures the domain plan being constructed to achieve the agents' shared goal, the problem-solving level which represents how the agents are going about constructing the domain plan, the belief level which captures the mutual beliefs being pursued to enable the problem-solving intentions, and the discourse level which represents the discourse actions being initiated to establish the mutual beliefs [4]. This paper focuses on the evaluation and modification of proposed beliefs.

The belief level of the dialogue model consists of mutual beliefs proposed by the agents' discourse actions. When an agent proposes a belief and provides (optional) supporting evidence for it, this set of beliefs is represented as a belief tree where the belief represented by a child node is intended to provide support for the belief represented by its parent node. Given a proposed belief tree, CORE determines its acceptance of the proposal based on its beliefs about the domain and about the user's beliefs. Associated with each belief is a strength that represents the agent's confidence in that belief correctly reflecting situations in the real world. The strength of a belief is modeled using endorsements, which are explicit records of factors that affect one's uncertainty in a hypothesis [6], following Logan et al. [9, 15]. These endorsements are based on the level of expertise of the agent conveying the belief, the surface form of the utterance used to convey the belief, stereotypical information, etc.

Figure 1 shows my algorithm for evaluating a proposed belief. Given a proposed belief tree, CORE invokes Evaluate-Belief with .bel instantiated as the belief represented by the root node of the tree. Since CORE's acceptance of .bel may be affected by its acceptance of the evidence that the user proposed to support it, the subtree that contains evidence supporting .bel in the proposed belief tree must be evaluated as part of evaluating .bel (step 3). A piece of evidence consists of a belief, .beli, and an evidential relationship, supporti(.beli,.bel). To determine the acceptance of a piece of evidence, CORE evaluates both the belief (step 3.1) and the evidential relationship (step 3.2) by recursively invoking Evaluate-Belief on them. A piece of evidence for .bel is accepted if both the child belief and the evidential relationship are accepted, and rejected otherwise.

To determine the acceptance of .bel, CORE constructs an evidence set (.evid-set) that contains the user's proposal of .bel, endorsed according to the user's level of expertise in that subarea as well as the strength of his belief as conveyed by the surface form of the utterance (step 1), CORE's own beliefs pertaining to .bel (step 1), and evidence proposed by the user that is accepted by CORE (step 3). The Estimate-Strength function is then invoked on .bel and .evid-set to determine whether or not .bel should be accepted based on the given evidence (step 4).

The Estimate-Strength function utilizes a simplified version of Galliers' belief revision mechanism [9, 15] to determine the acceptance of .bel by comparing the strengths of the pieces of evidence that support and attack .bel. Following Walker's weakest link assumption [30], the strength of a piece of evidence is the weakest of the strength of the belief, the strength of the evidential relationship, and the degree of evidence provided by the evidential relationship. Estimate-Strength then compares the strengths of the pieces of evidence for .bel and those again .bel to determine whether CORE believes .bel, ¬.bel, or has no belief about .bel. A proposal is accepted if the beliefs represented by the root nodes of the proposed belief trees (the top-level proposed beliefs) are accepted, and

1The degree of evidence provided by an evidential relationship measures the amount of support the antecedent belief provides for the consequent belief. For instance, a professor being on sabbatical may provide very strong support for him not teaching a course, while a professor's expertise being in algorithms may only provide strong support for him not teaching compilers. On the other hand, the strength of an evidential relationship measures the agent's confidence in the evidential relationship being true in the real world.

2This algorithm assumes that an agent may only accept or reject a proposed belief. However, my analysis of naturally occurring collaborative planning discussions shows that sometimes an agent does not have sufficient knowledge to determine whether to accept or reject a proposal, and will initiate an information-sharing subdialogue to exchange pertinent information with the other agent and subsequently re-evaluate the proposal. Discussions on the information-sharing process can be found in [5].
rejected otherwise. This is because the top-level proposed beliefs are the proposed beliefs that directly affect the problem-solving actions (and hence the domain plan) that the agents are proposing to perform; thus if these beliefs are agreed upon by both agents, it is irrelevant whether or not they agree on the evidence proposed to support them [35].

3.1 Example

To illustrate the evaluation process, consider the following utterances:

(8) S: You should take CS821 to satisfy your seminar course requirement.
(9) U: CS848 is a better course than CS821.
(10) The professor of CS848 is Dr. Lewis.

Figure 2 shows the belief level of the dialogue model inferred from utterances (9) and (10). CORE evaluates the proposal by invoking the Evaluate-Belief algorithm on the belief represented by the root node of the belief tree, Better-Than(CS848,CS821). Since evaluating Better-Than(CS848,CS821) requires that the evidence proposed to support it first be evaluated, CORE evaluates both Professor(CS848,Dr. Lewis) and supports(Professor(CS848,Dr. Lewis),Better-Than(CS848,CS821)). CORE gathers its private evidence pertaining to Professor(CS848,Dr. Lewis), which includes: 1) a very strong belief that Dr. Lewis is going on sabbatical in 1996 and a very strong belief that Dr. Lewis going on sabbatical provides very strong support for him not being the professor of a course (in this case, CS848), and 2) a strong belief that CS848 meets at 8am, a very strong belief that Dr. Lewis commutes 65 miles to work, and a very strong belief that these two beliefs provide strong support for Dr. Lewis not being the professor of CS848. CORE then invokes Estimate-Strength to determine its belief about Professor(CS848,Dr. Lewis) based on CORE’s own evidence and the user’s statement. Since CORE has one piece of very strong evidence and one piece of strong evidence against Professor(CS848,Dr. Lewis), and its only evidence for the belief is a strong piece of evidence conveyed by the user’s utterance, CORE rejects Professor(CS848,Dr. Lewis).

CORE believes that students generally prefer courses taught by good professors and that Dr. Lewis is a good professor; thus the proposed evidential relationship is accepted. However, CORE does not accept the top-level proposed belief, Better-Than(CS848,CS821) because it has a prior belief to the contrary (implicitly conveyed in utterance (8)) and the only evidence provided by the user was an implication whose antecedent was not accepted.

4 Modifying Rejected Proposals

The collaborative planning principle in [33, 30] suggests that “conversants must provide evidence of a detected discrepancy in belief as soon as possible.” Thus, once an agent detects a relevant conflict in a proposal, she must notify the other agent of the conflict and initiate a negotiation subdialogue to resolve it — to do otherwise is to fail in her responsibility as a collaborative agent. This process is captured by the Modify action in the Propose-Evaluate-Modify framework that attempts to modify a rejected proposal to a form that will potentially be accepted by both agents.

In CORE, the attempt to resolve a conflict and modify a rejected proposal is captured by the problem-solving action Modify-Proposal. When applied to belief modification, Modify-Proposal has two specializations: Correct-Node, applicable when a proposed belief is rejected, and Correct-Relation, appropriate when a proposed evidential relationship is rejected. Figure 3 shows the problem-solving recipes\(^\dagger\) for Correct-Node and one of its subactions, Modify-Node, that performs the actual modification of the proposal. The applicability conditions\(^\ddagger\) of Correct-Node specifies that the action can be pursued only when \(s_1\) believes that \(\_\text{node}\) is not acceptable while \(s_2\) believes that \(\_\text{node}\) is acceptable, i.e., when \(s_1\) and \(s_2\) disagree about the acceptance of the belief represented by \(\_\text{node}\). However, since this is a collaborative interaction, an agent cannot modify a proposal without the other agent’s consent; thus, before the actual modification of the proposal, the conflict between the agents must have been resolved. This is captured by the precondition of Modify-Node, the first subaction of Correct-Node. The attempt to satisfy this precondi-

\(\dagger\) A recipe [21] is a template for performing an action. It contains the applicability conditions for performing the action, the subactions comprising the body of the action, etc.

\(\ddagger\) Applicability conditions are conditions that must already be satisfied in order for an action to be reasonable to pursue, whereas an agent can try to achieve unsatisfied preconditions.
tion causes .s1 to attempt to establish the mutual belief that .node is not acceptable, leading her to adopt discourse actions to change .s2’s beliefs, thus initiating a collaborative negotiation subdialogue.

4.1 SELECTING FOCUS OF MODIFICATION

When an agent rejects a proposal, the agent may have rejected not only the top-level proposed belief(s), but also some pieces of evidence proposed to support it. In order to resolve the agents’ conflict about a rejected top-level proposed belief (.bel), the agent can directly provide evidence against .bel, or attack the rejected pieces of evidence, thereby eliminating the other agent’s reasons for holding .bel. Thus, in situations where multiple conflicts are detected, the agent must select the focus of modification — the aspect of the proposal which she will explicitly address in resolving the conflict about .bel. Since a collaborative agent is expected to engage in effective and efficient dialogues, she should address the rejected belief(s) that she predicts will most quickly and effectively resolve the top-level conflict.

My process for selecting the focus of modification involves two steps: 1) identifying the candidate foci tree, and 2) selecting a subset of the beliefs in the candidate foci tree to explicitly refute. The candidate foci tree contains the rejected pieces of evidence in the proposed belief tree which, if successfully refuted, have the potential to resolve the agents’ conflict about .bel. It is constructed by performing a depth-first search on the proposed belief tree. When a node is visited, both the belief and the evidential relationship between it and its parent are examined. If both the belief and relationship were accepted during the evaluation process, then search on the current branch terminates. This is because once a belief is accepted, it is irrelevant whether or not the evidence proposed to support it is accepted [35]. Otherwise, this piece of evidence is included in the candidate foci tree and CORE will continue to search through the evidence proposed to support the rejected belief/relationship.

Once the candidate foci tree is identified, CORE selects a subset of the rejected beliefs/relationships in the candidate foci tree that it will explicit refute by applying the Select-Focus-Modification algorithm to the root node of the candidate foci tree. Select-Focus-Modification, shown in Figure 4, analyzes .bel and its descendents and annotates .bel with its focus of modification (.bel.focus) — a set of beliefs/relationships that CORE will explicitly attack in order to resolve the agents’ conflict about .bel.

In step 1 of the algorithm, CORE gathers its beliefs about the user’s evidence pertaining to .bel (.bel.u-evid), as well as its own evidence against .bel (.bel.s-attack). If .bel is a leaf node in the candidate foci tree, i.e., there is no rejected evidence for .bel, then CORE

Select-Focus-Modification (.bel):
1. .bel.u-evid ← CORE’s beliefs about user’s evidence pertaining to .bel
   .bel.s-attack ← CORE’s own evidence against .bel
2. If .bel is a leaf node in the candidate foci tree,
   2.1 If PredICT (.bel, .bel.u-evid + .bel.s-attack) = ¬.bel,
      then .bel.focus ← { .bel } : return.
   2.2 Else .bel.focus ← nil : return.
3. Select focus for each of .bel’s children in the candidate foci tree, .bel1, . . . , beln:
   3.1 If .beli was rejected, Select-Focus-Modification (.beli).
   3.2 If .rel was rejected, Select-Focus-Modification (.rel).
4. Choose between attacking the proposed evidence for .bel and attacking .bel itself:
   4.1 .cand-set ← { .beli, .rel } \ rejected( .beli, .rel ) \ ¬l ∧ ¬rejeted .reli \ \∧ 
      \ v .bel1, .reli \ \∧ 
      \ v .reli, .reli \ \∧ 
   4.2 // Check if addressing .bel’s rejected evidence is sufficient
      If PredICT (.bel, .bel.u-evid - .cand-set) = ¬.bel,
      min-set ← Select-Min-Set (.bel, .cand-set)
      .bel.focus ← U .beli .min-set .beli.focus
   4.3 // Check if addressing .bel itself is sufficient
      Else if PredICT (.bel, .bel.u-evid + .bel.s-attack) = ¬.bel,
      .bel.focus ← { .bel }
   4.4 // Check if addressing both .bel and its rejected evidence is sufficient
      Else if PredICT (.bel, .bel.s-attack + .bel.u-evid - .cand-set) = ¬.bel,
      min-set ← Select-Min-Set (.bel, .cand-set ∪ .bel)
      .bel.focus ← U .beli .min-set .beli.focus ∪ (.bel)
   4.5 Else .bel.focus ← nil

Figure 4: Selecting the Focus of Modification
will invoke the `Predict` function to determine whether or not presenting the pieces of evidence in `_bel`'s attack will be sufficient to convince the user of `¬_bel`. If so, the focus of modification is `_bel`; otherwise, the focus of modification is nil, indicating that CORE does not have sufficient evidence to refute `_bel` (step 2). The `Predict` function is similar to `Estimate-Strength` in Section 3. However, instead of the evidence set being CORE’s evidence pertaining to `_bel`, it is CORE’s beliefs about the user’s evidence pertaining to `_bel` and CORE’s own evidence against `_bel` that CORE is planning on presenting to the user. Given this evidence set, the `Predict` function then predicts whether the user will believe in `_bel` or `¬_bel` after CORE’s evidence is presented to him, and returns the appropriate belief.

If `_bel` is not a leaf node in the candidate foci tree, then CORE must determine whether it will attempt to refute `_bel` by directly providing evidence against `_bel` itself, or addressing `_bel`’s rejected evidence to eliminate the user’s reasons for holding `_bel`. Thus CORE must first predict whether or not it has sufficient evidence to refute the rejected pieces of evidence that the user proposed to directly support `_bel`. If so, CORE must also determine how it will go about refuting them, i.e., determine the focus of modification for each such rejected belief. This is captured in step 3 of the algorithm in which CORE recursively applies the `Select-Focus-Modification` algorithm to the rejected belief (`_bel`) or evidential relationship (support(_bel,`_rel`)) of each piece of evidence that directly provides support for `_bel`. This recursive process annotates each rejected belief/relationship with its focus of modification (`.bel`, `.focus`, `.rel`, `.focus`) to indicate how CORE will go about refuting the belief/relationship. The value of `.bel`, `.focus` will be nil if CORE predicts that sufficient evidence is available to change the user’s belief about `_bel`. Based on this information, CORE then determines the most effective way to refute `_bel` (step 4) by selecting among attacking the user’s support for `_bel`, attacking `_bel` itself, or both.

CORE’s preference is to address `_bel`’s rejected evidence because McKeown’s focusing rules suggest that continuing a newly introduced topic is preferable to returning to a previous topic [18]. Thus CORE first considers whether or not attacking the user’s support for `_bel` is sufficient to convince him of `¬_bel`. It does so by first constructing a candidate set (.candset) which includes all pieces of rejected evidence proposed to directly support `_bel` which CORE predicts it can successfully refute. CORE then removes the candidate set from its beliefs about the user’s evidence pertaining to `_bel`, i.e., hypothesizing that the user now rejects each piece of proposed evidence in the candidate set, and invokes the `Predict` function on `_bel` with the remaining evidence (step 4.2). If CORE predicts that the user will now disbelieve `_bel`, indicating that refuting all evidence in the candidate set is sufficient to convince the user of `¬_bel`, CORE selects a minimum subset of the candidate set as the rejected beliefs it will actually refute. The focus of modification for `_bel` is then the union of the focus of modification for each belief/relationship selected in this minimum set.

If CORE predicts that attacking the evidence for `_bel` will not convince the user of `¬_bel`, it predicts whether or not directly attacking `_bel` will accomplish this goal. CORE combines its evidence against `_bel` with its beliefs about the user’s evidence pertaining to `_bel`, and invokes `Predict` with this set of evidence. If the user is predicted to accept `¬_bel`, then the focus of modification is `_bel` itself (step 4.3). If directly attacking `_bel` is again predicted to fail, CORE considers attacking both `_bel` and its supporting evidence by combining the previous two prediction processes (step 4.4). If the combined evidence is still predicted to fail, the CORE does not have sufficient evidence to change the user’s view of `_bel`; thus the focus of modification is nil.

After the `Select-Focus-Modification` process is completed, each rejected top-level proposed belief will be annotated with a set of beliefs on which CORE should focus when attempting to change the user’s view of `_bel`. The negations of these beliefs are then posted by CORE as mutual beliefs to be achieved in order to carry out the modification process. The next section discusses how CORE selects appropriate justification for each of these intended mutual beliefs.

### 4.2 Selecting Justification

Studies in communication and social psychology have shown that evidence improves the persuasiveness of a message [16, 25, 20, 13]. Research on the quantity of evidence indicates that there is no optimal amount of evidence, but that the use of high-quality evidence is consistent with persuasive effects [24]. On the other hand, Grice’s maxim of quantity [10] argues that one should not contribute more information than is required. Thus it is important that a collaborative agent select sufficient and effective, but not excessive, evidence to justify an intended mutual belief.

Given an intended mutual belief (.mb), CORE identifies the evidence that it will present to the user as support for .mb by applying the `Select-Justification`
Select-Justification(_mb):

1. _mb-evid ← CORE's beliefs about the user's evidence pertaining to _mb.
2. If Predict(_mb, _mb-evid + CORE's claim of _mb) = _mb, return _mb.
3. // Construct justification chains for _mb
   _mb-evidence ← CORE's evidence for _mb
   _evid-set ← {} For each piece of evidence in _mb-evidence, { _bel_i, _rel_i }:
   3.1 _bel_i-chain ← Select-Justification(_bel_i)
   3.2 _rel_i-chain ← Select-Justification(_rel_i)
   3.3 _evid-set ← _evid-set \cup Make-Evidence({ _bel_i-chain, _rel_i-chain }... _mb)
4. // Select justification chains that are strong enough to convince the user of _mb
   If _evid-set = {}, return nil.
   _set-size ← 1; _selected-set ← {}.
   _cand-set ← the set of all sets of justification chains constructed from _evid-set such that each element in _cand-set contains _set-size elements. For each element in _cand-set, _cand_1, ... _cand_n:
   4.3.1 If Predict(_mb, _mb-evid + CORE's claim of _mb + _cand_i) = _mb,
         _selected-set ← _selected-set \cup { _cand_i }
4.4 If _selected-set = {}, _set-size ← _set-size + 1 If _set-size ≤ number of elements in _evid-set,
         goto step 4.3;
   Else return nil.
5. _selected-set ← evidence in _selected-set about which CORE is most confident
6. _selected-set ← evidence in _selected-set most novel to the user
7. _selected-set ← evidence in _selected-set that contains the fewest beliefs
8. Return first element in _selected-set

Figure 5: Selecting Justification for a Claim

algorithm (Figure 5) to _mb. For each intended mutual belief, CORE first predicts if merely presenting _mb will cause the user to accept the belief (step 2). If so, then no justification will be given; otherwise CORE will select appropriate justification by identifying the candidate justification chains that could be used to support _mb (steps 3 and 4) and applying filtering heuristics to them (steps 5 - 7).

CORE first gathers all the pieces of evidence that could be used to justify _mb in a set named _mb-evidence (step 3). The Select-Justification algorithm is then recursively invoked on the belief (_bel_i) and evidential relationship (_rel_i) that comprise each piece of evidence in _mb-evidence to determine the justification needed for each of them (steps 3.1 and 3.2). Thus _bel_i-chain and _rel_i-chain are justification chains that specify how CORE will go about convincing the user of _bel_i and _rel_i, respectively. Furthermore, since _bel_i and _rel_i together constitute a piece of evidence that supports _mb, CORE invokes Make-Evidence to construct a justification chain with _mb as its root node, the root node of _bel_i-chain (_bel_i) as its child node, and the root node of _rel_i-chain (_rel_i) as the relationship between _bel_i and _mb (step 3.3). This justification is then added to _evid-set, which contains alternative justification chains that CORE can present to the user as support for _mb. The recursive calls in this algorithm terminates when 1) CORE finds a belief that the user already holds or is predicted to hold when the belief is presented to her, in which case no further justification is needed, or 2) CORE cannot find sufficient justification to convince the user of a belief necessary to establish _mb, in which case the search for a justification chain fails.

Once a set of candidate justification chains is identified (_evid-set), CORE selects from _evid-set the chains of justification which, when presented, are predicted to convince the user of _mb (step 4.3). If no single justification chain would accomplish this goal, CORE combines the justification chains until a large enough subset is found which is predicted to change the user’s view about _mb (step 4.4). This produces a set of candidate justification chains, and three heuristics are then applied in turn to select among them. The first heuristic prefers evidence in which CORE is most confident since high-quality evidence produces more attitude change than any other evidence form [16]. Furthermore, CORE can better justify a belief in which it has high confidence should questions about its validity arise. The second heuristic prefers evidence which is novel to the user, since studies have shown that evidence is most persuasive if it is previously unknown to the hearer [34, 19]. Finally, the third heuristic prefers evidence that contains the fewest beliefs, based on Grice’s maxim of quantity [10].

4.3 Example

To illustrate the modification process, we return to the example in Section 3.1. CORE evaluated the proposed belief tree in Figure 2 and rejected both Better-Than(CS848,CS821) and Professor(CS848,Levis). Since the top-level proposed belief is rejected, the Modify-Proposal action is invoked in an attempt to resolve the agents’ conflict and to modify the proposal to a form that will potentially be accepted by both agents.

The modification process requires that the focus of modification first be selected. Thus, CORE identifies the candidate foci tree which, in this
example, is identical to the proposed belief tree since both the top-level proposed belief and the evidence proposed to support it were rejected. CORE then applies the Select-Focus-Modification algorithm to the root node of the candidate foci tree (Better-Than(CS848,C8821)). The algorithm specifies that to determine the focus of modification for a belief, the focus of modification for each of the rejected piece of evidence must first be identified (step 3); thus the algorithm is recursively applied to the rejected child belief, Professor(CS848,Lewis). CORE has two pieces of evidence against Professor(CS848,Lewis): 1) a very strong piece of evidence which consists of On-Sabbatical(Lewis,1996) and supports(On-Sabbatical(Lewis,1996), ~Professor(CS848,Lewis)), and 2) a strong piece of evidence which consists of Meets-At(CS848,8am), Commutes(Lewis,65 miles), and supports(Meets-At(CS848,8am) ∧ Commutes(Lewis,65 miles), ~Professor(CS848,Lewis)). CORE predicts that these two pieces of evidence are sufficient to change the user’s belief in Professor(CS848,Lewis); thus the focus of modification for Professor(CS848,Lewis) is the belief itself. In determining the focus of modification for Better-Than(CS848,C8821), CORE’s preference is to address the rejected evidence. Since CORE predicts that eliminating the proposed piece of evidence is sufficient to change the user’s belief about Better-Than(CS848,C8821), the focus of modification for Better-Than(CS848,C8821) is Professor(CS848,Lewis). The Correct-Node action (Figure 3) is therefore selected as the specialization of Modify-Proposal, and MB(S,U,~Professor(CS848,Lewis)) is posted as an intended mutual belief in order to satisfy the precondition of Modify-Node.

Given the intended mutual belief, CORE invokes the Select-Justification algorithm (Figure 5) on the belief to determine the justification it will present to the user as support for the belief. Since CORE believes that the user has a very strong belief in Professor(CS848,Lewis) (conveyed by the surface form of utterance (10)), CORE predicts that merely informing the user that the professor of CS848 is not Dr. Lewis is not sufficient to convince the user of ~Professor(CS848,Lewis) (step 2). Thus, additional justification for the intended mutual belief is needed.

CORE constructs the alternative justification chains that can be used as support for ~Professor(CS848,Lewis) by recursively applying the Select-Justification algorithm to the two available pieces of evidence for the belief (step 3). This results in the two justification chains shown in Figure 6. CORE predicts that either piece of evidence combined with the proposed mutual belief is sufficient to change the user’s belief; therefore, the filtering heuristics are applied. The first heuristic prefers justification chains in which the system is most confident, causing the justification chain in Figure 6(a) to be selected. Thus, CORE generates the logical forms of the following utterances in an attempt to resolve the agents’ conflict detected in the user’s original proposal:

(11) S: The professor of CS848 is not Dr. Lewis.
(12) Dr. Lewis is going on sabbatical in 1996.
(13) Dr. Lewis was given tenure in 1995.

If the user accepts CORE’s utterances, thus satisfying the precondition that the conflict be resolved, Modify-Node can be performed and changes made to the user’s original proposal. Otherwise, the user may propose modifications to CORE’s proposed beliefs, resulting in an embedded negotiation subdialogue.

5 RELATED WORK

Researchers have studied the analysis and generation of arguments [2, 23, 7, 29, 22, 17]; however, agents engaging in argumentative dialogues are solely interested in winning an argument and thus exhibit different behavior from collaborative agents.

Webber and Joshi [32] have noted the importance of a cooperative system providing support for its responses. They identified strategies that a system can adopt in justifying its beliefs; however, they did not specify the criteria under which each of these strategies should be selected. Walker [31] described a
method of determining when to include optional warrants to justify a claim based on factors such as communication cost, inference cost, and cost of memory retrieval. However, her model focuses on determining when to include informationally redundant utterances, whereas my model determines whether or not justification is needed for a claim to be convincing and, if so, selects appropriate evidence from the system's private beliefs to support the claim.

Caswey et al. [3, 15] introduced the idea of utilizing a belief revision mechanism [9] to predict whether a set of evidence is sufficient to change a user's existing belief and to generate responses for information retrieval dialogues in a library domain. They argued that in the library dialogues they analyzed, "in no cases does negotiation extend beyond the initial belief conflict and its immediate resolution." [15, page 141]. However, my analysis of naturally-occurring consultation dialogues shows that in other domains conflict resolution does extend beyond a single exchange of conflicting beliefs; therefore I employ a recursive model for collaboration that captures extended negotiation and represents the structure of the discourse. Furthermore, their system deals with a single conflict, while my model selects a focus in its pursuit of conflict resolution when multiple conflicts arise. In addition, I provide a process for selecting among multiple possible pieces of evidence.

ACKNOWLEDGMENTS

I would like to thank Sandee Carberry, Candy Sider, Kathy McCoy, and Stephanie Elzer for helpful discussions and comments during the development of this work. I would also like to thank Rachel Sacher for her help in implementing CORE, and Mark Chu-Carroll for editorial comments.

REFERENCES


Interaction management functions and context representation requirements

Harry Bunt
Tilburg University, The Netherlands
bunt@kub.nl

1 INTRODUCTION

Communication is a complex multifaceted activity, various aspects of which must be managed by the participants in order be effective, efficient and pleasant. Linguistic and other communicative behavior that has a specific purpose in this respect, such as feedback utterances ("That's right", "OK", "What did you say?"), attention signals ("Hello?!", "Yes"), or turn-taking markers ("John?"), illustrate the fact that speakers often manage the interaction explicitly. In this paper we will pay special attention to communicative acts that have such management functions, but we will also be concerned with dialogue management in a broader sense, dealing in a rather fundamental way with the mechanisms that motivate communicative activity and drive a dialogue forward. A special focus in this paper is the consideration of what the analysis of dialogue management phenomena tells us about requirements for an adequate modelling of the context in dialogue systems, which we believe to be a necessary basis for building intelligent, efficient and pleasant interactive systems. We will approach these issues in the framework of a theory of dialogue called Dynamic Interpretation Theory (DIT), which analyses the meaning of communicative behaviour in terms of context change.

Approaching meaning in terms of context change offers the possibility to deal with dialogue utterances that seem impossible to interpret in more traditional, truth-based approaches. Utterances such as "Hello", "Good morning", "Airport Information Service", "Thank you", "OK" illustrate this. In addition, and more important for the purpose of the present paper, the context-change approach to utterance meaning also provides a basis for understanding dialogue mechanisms: viewing the intended meaning of an utterance to be the way the speaker wants to change the context, the motivation for performing the utterance is immediately seen to be some aspect of the context, which the speaker wants to change.

In order to make this approach work, the notion of context must be given a manageable interpretation. We try to obtain such a notion by investigating communicative activity in dialogues and analysing the aspects of context addressed by communicative acts. In the present paper we will do this for a subclass of communicative acts, so-called 'interaction management acts', which are specifically concerned with the management of aspects of the interaction. We will analyze the context aspects addressed by such acts and deduce requirements for context modelling. Our analysis will be backed up by both theoretical considerations provided by the DIT framework and by empirical data.

The rest of this paper is organized as follows. We first give a brief summary of the main traits and central assumptions of Dynamic Interpretation Theory as far as relevant to the main purpose of this paper (Section 2). This includes a general discussion of motivations and mechanisms of communicative activity. We then turn to the analysis of interaction management acts and requirements on context modelling (Section 3). We present a corpus-based inventory of IM functions, and analyse their conditions of use and their context-changing effects in the light of the DIT approach to structured context modelling.

2 DYNAMIC INTERPRETATION THEORY

Dynamic Interpretation Theory (DIT) has emerged from the study of spoken human-human information dialogues, and aims at uncovering fundamental principles in dialogue both for the
Dialogue 1. Human-human telephone information dialogue between client (C) and information service at Schiphol, Amsterdam Airport (I).

1 I: Schiphol information, good morning
2 C: good morning, this is De Bruin speaking
3 C: can you tell me which planes will leave for Frankfurt between twelve and three o'clock?
4 I: just a moment, please
5 I: hello?
6 C: yes
7 I: at twelve fifty-five the KLM will leave...
8 C: yes
9 I: the KL 243...
10 C: the KL 243
11 I: correct
12 I and at one a.m. the Garuda leaves...
13 C: yes
14 I: and it will make its first intermediate stop in Frankfurt
15 C: yes
16 I: and... between twelve and three you said?
17 C: yes
18 I: yes, and there is another one at thirteen thirty of Turkish Airlines.
19 C: Turkish Airlines?
20 I: yes
21 C: oh
22 I: and there is one more at exactly three o'clock of Phillipan Airlines
23 C: Phillipan Airlines?
24 I: yes
25 C: OK, thanks very much
26 I: you're welcome
27 C: goodbye
28 I: goodbye

purpose of understanding natural dialogue phenomena and for the purpose of designing effective, efficient and pleasant computer dialogue systems.

Information dialogues, which form the focus of these studies, serve the purpose of exchanging information concerning some domain of discourse. Such a task naturally gives rise to questions, answers, checks, confirmations, etc. In addition, information dialogues also contain other elements such as greetings, apologies, acknowledgements, farewells, etc. We refer to the first type of elements as task-oriented acts and to the latter as dialogue control acts. Task-oriented acts are directly motivated by the task or purpose underlying the dialogue and contribute to its accomplishment; dialogue control acts are concerned with the interaction itself, and serve to create and maintain the conditions for smooth and successful communication. The interaction management acts, on which this paper focuses, form a subclass of the dialogue control acts.

Dialogue 1 illustrates the use of task-oriented and dialogue control acts in information dialogues; it is a human-human dialogue over the telephone with the information service at Schiphol, Amsterdam Airport.¹

2.1 Dialogue acts

The idea to view communicative acts as context-changing operations has been suggested in the speech act literature (e.g., Gazdar, 1979; Isard, 1975), but has not been worked out to the point of formalization. In particular, Gazdar (1979) has suggested to formalize the concept of 'illocutionary force' in terms of context changes.

¹The Dutch original has been translated into English here.
In Bunt (1989) we have presented a formalized hierarchical system of task-oriented dialogue acts defined as context-changing operations, where context is construed as the pair consisting of the states of information of the two participants. At the top of this hierarchy we find two subclasses of dialogue acts, those concerned with information seeking and those with information providing. Dialogue control acts, which have received much less attention in the speech act literature, are discussed in Bunt (1994), where a classification is presented into three subsystems, concerned with feedback, interaction management, and social obligations management. Feedback acts provide information about the processing of partner inputs, reporting or resolving problems (negative feedback), or reporting success (positive feedback). Social obligations management acts deal with socially indicated obligations should as welcome greeting, thanking, apologizing for mistakes or inability to perform a function, and farewell greeting. Interaction management acts handle various aspects of the interactive situation, such as taking turns, pausing and resuming, and attention monitoring; these acts will be considered in more detail below. Figure 1 provides a schematic overview of these subsystems.

```
+-+ information seeking
   /|
  /  orientated + information providing
 ||
 |  +-- positive
 #| |
 ^|  +-- feedback --|
 | |  |  +-- negative
 | +-- dialogue --
 |   | interaction management
 |   | control
 |   / +-- social obligations management
```

Figure 1. Subsystems of communicative functions in information dialogues.

In the foregoing, we have used the terms 'dialogue act' and 'communicative act' in an informal way, assuming that the reader would interpret 'communicative act' as 'act of communicative behaviour', and 'dialogue act' as 'communicative act, used in the setting of a dialogue'. We will continue to use the term 'communicative act' in this sense, but we will give a more specific meaning to the term 'dialogue act'. 'Dialogue acts' are defined as the functional units used by the speaker to change the context (Bunt, 1994).

These functional units do not correspond to natural language utterances, sentences, or other linguistic units in a simple way, because utterances are often multifunctional, as we will see below. The importance of the stipulation ... used by speakers is that we are considering context update functions that are elementary in the sense that they cannot be defined in terms of the update operations of other, more primitive dialogue acts; every communicative function thus corresponds to particular (combinations of) features of observable communicative behaviour.

A dialogue act has a semantic content, formed by certain information that has a particular significance in the context, and a communicative function that defines the significance of this information by specifying in what way the information should be used to change the context. Formally, a communicative function is a mathematical function that, given a semantic content as its argument, operates on a context and produces a new context. A dialogue act is linked to observable communicative behaviour through its 'utterance form'.

Using the term 'utterance' to refer to everything contributed by a speaker in one turn, an utterance may correspond to more than one dialogue act, and thus be multifunctional, for several reasons. First, an utterance may consist of several sentences or phrases that each express dialogue acts. So dialogue acts often relate to parts of utterances. Moreover, utterances or utterance parts often carry more than one functional meaning, because of (1) indirectness: a question like "Do you know the arrival time?" may function indirectly as a request to tell the arrival time; (2) 'functional subsumption': a promise like "I will come tonight" is, besides a promise, also an informative statement; (3) 'functional multidimensionality': dealing with the underlying task is very often combined in one utterance with dialogue control aspects; for example, an answer to a question also offers feedback information, since it implicitly indicates that the question was understood and accepted.

2.2 Dimensions of Context

As mentioned above, utterance meaning is defined in DIT in terms of context change. In the literature, the term 'context' is used in many different ways, referring for example to the preced-

---

5On multifunctionality see also Allwood et al. (1989; 1992; 1994).
ling discourse, to the physical environment, or to the domain of discourse. These uses all have in common that they refer to factors, relevant to the understanding of communicative behaviour. In Bunt (1994) we have argued that these factors can be grouped into five categories: cognitive, semantic, physical, social, and linguistic. Moreover, for each of these ‘dimensions’ of context we may fruitfully distinguish between global aspects, which are fixed at the beginning of the dialogue and remain constant throughout, and local aspects, whose values develop and change through the dialogue. These categories of context factors may be characterized briefly as follows.

- **Linguistic context**: surrounding linguistic material, ‘raw’ as well as analysed. Closely related to what is sometimes called ‘Dialogue History’ (see e.g. Bilange, 1991; Pernel and Prince, 1995).

- **Semantic context**: state of the underlying task; facts in the task domain.

- **Cognitive context**: participants’ states of processing and models of each other’s states.

- **Physical and perceptual context**: availability of communicative and perceptual channels; partners’ presence and attention.

- **Social context**: communicative rights, obligations and constraints of each participant.

For identifying a communicative function we have two criteria, that follow immediately from the definition of a dialogue act:

- the function defines a specific way of changing the context, which is elementary in the sense that this context change cannot be obtained through a combination of dialogue acts with other communicative functions;

- specific features of communicative behaviour are used by speakers to indicate this function.

Applying these criteria to task-oriented dialogue acts, classifications of TO-functions and DC-functions were obtained in Bunt (1989) and Bunt (1994), respectively, of which the top level was shown in figure 1. Below we will consider the subsystem of interaction management functions in some detail.

### 2.3 Why Communicative Agents Act

In the opening line of this paper, we introduced the managing of communicative activity as serving to achieve an effective, efficient and pleasant interaction. Effectiveness, efficiency and pleasantness correspond to three basic assumptions in DIT:

1. People communicate in order to achieve something. Communication, in other words, is assumed to serve certain goals or purposes, often not communicative in nature, for which the communication is instrumental; in that respect it is supposed to be effective.

2. In communication, the instruments offered by language and other media (gestures, mimics, direction of gaze,...) are used to achieve the underlying goals and purposes. Communicative behaviour is assumed to be relevant in this respect; in avoiding irrelevant activity, the participants act efficiently.

3. Communication between people is a form of social interaction. One is not supposed to ignore the goals and limitations of the dialogue partner, nor to make the communication deliberately difficult or otherwise unpleasant (unless that would in itself serve a goal). Taking the partner’s interests into account can also be described as being cooperative.
Saying that a dialogue partner has the desire to be effective, efficient and cooperative, points in the direction of the fundamental forces that drive the communicative behaviour of a dialogue participant. We can describe these forces as follows:

**Rationality** Participants form communicative goals in accordance with underlying goals and desires, choose their actions so as to further their communicative goals and satisfy their desires, and try to make the communication successful.

**Cooperativity** Participants take the partner’s goals into account, choosing their actions so as to further these goals.

**Sociality** Partners act according to the norms and conventions for pleasant and comfortable interaction.

These assumptions lead to the following sources of motivation for communicative activity:

1. Communicative goals motivated by goals of the task, for the performance of which the dialogue is meant to be instrumental. The rationality explains hypothesis predicts this.

2. Recognized partner goals. According to the cooperativity hypothesis, recognition of such goals may be sufficient reason for a participant to form the intention to act, as opposed to other theories where a cooperative agent must adopt partner goals.

3. Uncertainties or actual or anticipated problems in the conditions for successful communication. When a dialogue participant notices a problem or is afraid a problem might arise, he is expected, by to the rationality hypothesis, to perform communicative actions to improve the conditions.

4. Social obligations, such as greeting, apologizing, and thanking.

Schematically, we may represent the various ways in which a communicative agent may be motivated to perform a communicative act as follows:

\[
\begin{align*}
\text{if} & \quad \text{Agent } X \text{ has goal } P \\
& \quad \text{Act } \alpha \text{ could realize } P \\
or & \quad \text{Agent } X \text{ knows that } Y \text{ has goal } P \\
& \quad \text{Act } \alpha \text{ could realize } P \\
or & \quad \text{Agent } X \text{ notices (potential) communicative problem } P \\
& \quad \text{Act } \alpha \text{ could signal or resolve } P \\
or & \quad \text{Agent } X \text{ is under social obligation } P \\
& \quad \text{Act } \alpha \text{ could resolve } P \\
\text{then} & \quad \text{Agent } X \text{ is motivated to perform } \alpha
\end{align*}
\]

**Figure 2. Motivations for communicative action.**

The different motivations of communicative action relate to the subsystems of communicative functions, shown in figure 1, as follows.

**Task-oriented**: For information dialogues where one of the participants is seeking information, as in all cases where information services or information systems are involved, the acts with a \( TO/\text{information-seeking} \) function and a semantic content related to the task are driven by the goals of the task. The acts with a \( TO/\text{information-providing} \) function and a semantic content related to the task are driven by the Cooperativity principle.

**Dialogue control**: \( \text{Positive feedback} \) acts are motivated by a social principle which requires feedback from time to time (see below). \( \text{Negative feedback} \) is motivated, through the Rationality principle, by the drive to communicate successfully; this requires any obstacles in this respect to be removed. Similarly for \( \text{interaction management} \) acts (but see below).

\( \text{Social obligations management} \) acts are motivated by the desire to honour social obligations.

### 2.4 Context Modelling

To apply the context-change approach of DIT to computer dialogue system design, the maintenance of an articulate context model in the machine is essential. In Bunt (1989) we construed local context models for task-oriented dialogue acts as pairs \(< K_A, K_B >\), where \( K_A \) is the state of knowledge, belief and intentions of partner \( A \), and \( K_B \) that of \( B \). This approach has been implemented in the TENDUM dialogue system (Bunt et al., 1985). In Bunt (1995b) we proposed a richer notion of context to also handle dialogue control acts, where local context models get the
form $< C_A, C_B >$, $C_A$ being the local context according to $A$, consisting of the 5 dimensions mentioned above, and $C_B$ that according to $B$. Within the local cognitive context, moreover, two components are distinguished: the system's processing status and the system's model of the user's model of the local context. This latter part is needed since any time the system performs a DC act concerning some aspect of local context, such as the local linguistic context (as in "Please repeat"), it makes an assumption about the user's linguistic context (in this example that the user has stored his previous utterance). This is more generally true: whenever a dialogue act arises due to some property of information of type $T$ in the speaker's context, some assumptions are made about the information of type $T$ in the addressee's context. In other words, for every aspect of $C_A$ which is relevant, also $A$'s model of the same aspect of $C_B$ is relevant; this leads to a recursion in the model structure. Using $S$ and $U$ for system and user, we thus have, for the system's model of local context:

<table>
<thead>
<tr>
<th>$C_S$</th>
<th>$&lt; S$'s loc. semantic context,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S$'s loc. cognitive context:</td>
</tr>
<tr>
<td></td>
<td>$&lt; $ Own Proc. State $</td>
</tr>
<tr>
<td></td>
<td>$User Model: C_{SU} &gt;$</td>
</tr>
<tr>
<td></td>
<td>$S$'s loc. physical and</td>
</tr>
<tr>
<td></td>
<td>perceptual context,</td>
</tr>
<tr>
<td></td>
<td>$S$'s loc. social context,</td>
</tr>
<tr>
<td></td>
<td>$S$'s loc. linguistic context &gt;</td>
</tr>
</tbody>
</table>

*Figure 3. The structure of local context.*

Here, $C_{SU}$ is used to indicate the local context as the system believes the user views it.

Participants in a dialogue obtain information about each other's view of the local context by interpreting each other's communicative actions. Successful communication, with correct understanding of a dialogue act $A_a$, creates mutual beliefs concerning the local context aspects addressed by $A_a$, where different subsystems of communicative functions address different context aspects.

We already noticed that all communicative functions address the local cognitive context, and that TO-acts also address the semantic context. Social obligations management acts address the local social context, in a way that can be described with the help of 'interactive pressure principles', describing how properties of the local context create pressures on dialogue participants to perform certain communicative acts like thanking and apologizing, and reactive pressure properties that describe the strictly local creation of pressures to react to such acts in particular ways, thereby explaining the phenomenon commonly known as 'adjacency pairs' of speech acts (see further Bunt, 1995).

Feedback acts address the state of processing of the speaker and the addressee's information about that, which form part of the local cognitive context. Negative feedback acts, as we have already seen, address problems in the processing of partner inputs, and are rationally motivated by the desire to communicate successfully; positive feedback acts are perhaps best seen as partly rationally motivated by the desire to remove any doubt about the success of communication, and partly socially motivated by an interactive pressure principle requiring to signal every now and then that attention is paid to the speaker. (See also Bunt, 1995, where a distinction is made between 'auto-feedback' and 'allo-feedback', that we do not address here.)

Interaction management acts are concerned with various aspects of the interactive situation, and do not address one particular dimension of local context. We now turn to the analysis of interaction management acts and the requirements that follow for context modelling.

### 3 Interaction Management Acts and Context Representation

#### 3.1 IM Functions

The exploration of corpora of spoken information dialogues shows that interaction management acts in this kind of dialogue focus on turn-taking, timing, contact, dialogue structuring, and the utterance formulation process. More specifically, in telephone information-seeking dialogues between an information service and clients calling up the service, we have identified the Interaction Management (IM) functions displayed in figure 2, where capital letters indicate communicative functions and small letters indicate classes of communicative functions. Table 1 provides a number of examples of how these functions are expressed in Dutch and in English.

Dialogue acts concerned with 'own communication management (OCM)', a term introduced by Allwood (1989), serve to indicate obstacles in a speaker's utterance formulation process.
The two cases of OCM we have identified are retraction, where the speaker signals that he has made an error and retracts something, and self-correction, in which case the speaker replaces some erroneously produced material by something else. It is often necessary for an addressee to be aware of OCM acts, as he would otherwise be likely to misunderstand the speaker.

'Time management' acts are concerned with activities that the speaker is planning. A pause occurs when the speaker estimates that the time he needs is too long for an unexplained silence. When he thinks relatively little time is needed, he may instead protract, typically combined with a turn keeping act ("Eh").

Contact management acts frequently occur in telephone dialogues when a speaker is uncertain whether the person at the other end of the line is actually there and is paying attention, especially after a pause. "Hello?" is commonly used to check this; "Yes" is the prototypical confirmation of presence and attention.

Turn management in conversation has been studied extensively in the tradition of conversation analysis (see Sacks, 1992; Schegloff, 1986; Sacks, Schegloff and Jefferson, 1974). In the dialogues we studied the most important cases of turn management are the following:

1. The information service person encourages the client to go on, in combination with positive feedback. This happens especially in the early part of a dialogue, where the client is formulating his request for information, as in the following example:

   I: Airport Information.
   C: Good morning, with Van den Broek, I wanted to ask you something about flights to Toronto, eh...
   I: Yes...

   Here the last "Yes" is spoken in a prolonged fashion with an intonation that first goes down and then rises sharply. This exemplifies turn-giving.

2. The speaker speaker needs a little time for his own communication management or for information gathering (protraction).

3. The speaker is interrupted because an error is detected that should preferably be addressed right away (interruption).
<table>
<thead>
<tr>
<th>IM function</th>
<th>communicative devices Dutch/general</th>
<th>communicative devices English/general</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>own comm. man.:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>retraction</td>
<td>“of nee...”, “ik bedoel...”</td>
<td>“I mean...”</td>
</tr>
<tr>
<td>self-correction</td>
<td>phrase substitution</td>
<td></td>
</tr>
<tr>
<td><strong>time management:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>protraction</td>
<td>“Eh...”, “Even kijken...”</td>
<td>“Eh”; “Let me see”</td>
</tr>
<tr>
<td>pause</td>
<td>decreasing speech time</td>
<td>“Just a moment”</td>
</tr>
<tr>
<td>“Een ogenblikje”</td>
<td></td>
<td>“Let me have a look”</td>
</tr>
<tr>
<td>“Ik zal even voor u kijken”</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>turn management:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>turn giving</td>
<td>“Ja”</td>
<td>“Yes”</td>
</tr>
<tr>
<td>turn keeping</td>
<td>rising final intonation</td>
<td>slow repetition of phrase</td>
</tr>
<tr>
<td>interruption</td>
<td>“Wacht”</td>
<td>“Wait”</td>
</tr>
<tr>
<td>hand raising gesture</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>contact management:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pres./att. indication</td>
<td>“Hallo!”</td>
<td>“Hello!”</td>
</tr>
<tr>
<td>eye contact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pres./att. check</td>
<td>“Hallo?”</td>
<td>“Hello?”</td>
</tr>
<tr>
<td>pres./att. request</td>
<td>“Juffrouw”, “Luister”</td>
<td>“Miss!”, “Listen”</td>
</tr>
<tr>
<td>pres./att. confirmation</td>
<td>“Ja”</td>
<td>“Yes”</td>
</tr>
<tr>
<td>dialog act announcement</td>
<td>“Vraagje”</td>
<td>“A little question.”</td>
</tr>
<tr>
<td><strong>topic management:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>topic indication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP fronting; pointing</td>
<td>“Jets anders”, “En nog iets”</td>
<td>“Something else”; “Also...”</td>
</tr>
<tr>
<td>shift announcement</td>
<td>dislocated NPs and PPs</td>
<td></td>
</tr>
<tr>
<td>change indication</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Interaction management functions and communicative devices

**Discourse structuring acts** are those acts which a speaker performs in order to structure the interaction by indicating that he is closing the discussion of a certain topic, that he wants to address a new topic, or that he indicates in some other way that he wants to structure the dialogue in a certain fashion. A variety of syntactic and prosodic instruments for explicitly introducing new topics or reintroducing previous topics are used (see Rats, 1996 for a study of topic management in information dialogues). Discourse structuring acts are based on the speaker’s view of the present linguistic context and on his plan for continuing the dialogue, and on assumed lack of clarity of the topical structure of the linguistic context for his partner.

It should be noted that the specific inventory of IM functions described here applies to the situation of information-seeking dialogues over the telephone with an information service in the Netherlands. It is well known that the details of inventories of this kind depend on a lot of linguistic, technical, and socio-cultural factors, such as whether the interaction is face-to-face or via telephones; whether both interactants are humans or one of them is a machine; whether they use speech or keyboard/screen, what linguistic community (or communities) they belong to, etc.

### 3.2 IM CONTEXT CONDITIONS

For each of the IM functions distinguished above we can formulate the conditions on local context that may give rise to their use, and the corresponding changes they bring about, when used successfully - these changes are, in general, simply the beliefs expressing the recognition that the context conditions hold for the speaker. We will do this here for a subset of the IM functions, which is complete in the sense that it involves all the context aspects that play a role in the entire system of IM functions.
SELF-CORRECTION: the speaker has produced a (partial) result that he recognizes to be wrong

PAUSE: the speaker estimates that he needs to do some off-line processing that takes substantial time, before being able to continue the interaction

TURN GIVING: one of the partner’s goals has just been satisfied; or the partner has just come in a position so as to act with respect to a speaker goal; or there is no motivation for the speaker (other than this one) to perform a communicative act

PRES/ATT CHECK: the interaction has been suspended; the speaker wishes to resume the interaction, and is uncertain as to whether the partner is present and paying attention

TOPIC CHANGE INDICATION: the speaker plans his next TO-act to be about an entirely different topic than the previous part of the dialogue

This tells us what aspects of the interactive state should be modelled in order to interpret and to generate IM acts; moreover, an analysis of the propositional contents of IM acts tells us in what detail these aspects have to be represented. Focusing on IM acts that use specific IM devices, an interesting point may be noticed here: virtually all such IM acts have very little internal structure in their content! Time management acts, for instance, mostly indicate only that some extra time is needed; at most, they specify the activity that needs time (“One moment please, I will look that up for you”). These activity names are also needed for feedback acts; the additional aspect entering here is the time estimated for an activity. This time estimate is never articulate. Time management thus requires only one extra parameter per process in the modelling of its state of processing, with 3 possible values: ‘negligible’, ‘small’ (for PROTRACTION) and ‘substantial’ (for PAUSE).

OCM acts to not require any new kind of context information to be modelled, beyond that which is already needed for feedback purposes. Negative feedback acts report or correct problems in processing partner inputs; OCM acts do the same for output generation processes. The extension required by OCM is thus that output generation processes should be among the processes whose status is represented in the Processing State part of the local cognitive context.

Without going into a detailed analysis of feedback here, we may assume that feedback, OCM and time management essentially require a representation of process information that can be realised by means of a small number of parameters per process, as shown in attribute-value matrix form in figure 5. The RESULT parameter is intended to have a list of values R4 which may be complex structures, like nested feature structures.

Contact management acts we have already seen to be rather trivial, from the point of view of context modelling. Two parameters, one for the speaker’s presence & attention and one for the partner’s (as believed by the speaker), are sufficient. These parameters should be part of the local Physical/Perceptual Context representation.

The context information required for turn management is perhaps best treated as integrated with linguistic context. Linguistic context is usually viewed as a record of the ‘linguistic history’, i.e. a record of the linguistic behaviour so far. This is also known as ‘dialogue history’ (see Prince and Pernc, 1996). In order to take dialogue participants’ anticipations as to how the dialogue will continue into account, we also need a ‘planned dialogue future’, containing the same kind of information as the dialogue history, but with less detail. The turn-taking present and future can be represented simply by a parameter indicating the speaker of each contribution. The conditions motivating an INTERRUPTION are already expressed in the partner’s Process State representation.
Topic management requires certain elements in the linguistic context to be marked as having topical status. Our corpus investigations provide little evidence of elaborate communicative planning, reflected in discourse structure acts. As far as the planning of topics is concerned, speakers in these dialogues do not seem to do more than (1) planning an unordered set of topics to be addressed; (2) deciding on the first/next topic from this set. The representation of complex topical chains or even more complex structures does not seem necessary.

The various kinds of context information needed to account for the generation of IM acts all constitute information available to the speaker or, put differently, constitute beliefs of the speaker. For an addressee, the understanding of these acts requires the same kinds of information to be represented as the addressee believes them to be believed by the speaker. Representation of IM context information therefore needs to be nested, just like all other local context information. The recursive context structure shown above in figure 3 provides for arbitrarily deep nesting of all local context information, including IM information.

It might be objected that this structure leads to too much nesting for IM information. Indeed, turn-taking, timing, self-correction, contact, and discourse structure are never the subject of explicit communication, consistent with the fact that these acts mostly have a marginal propositional content. It therefore seems implausible that dialogue participants would build up elaborate recursive structures around this kind of information. An attractive alternative to the fully recursive context model may therefore be one where the recursion is restricted to beliefs relating to the semantic context, and where the other components have a limited finite nesting of only a few levels deep.

We noted above that IM acts nearly always have only a marginal semantic content, with very little internal structure; their meaning is largely concentrated in their communicative function. This suggests that the representation of the relevant context information does not require a powerful, expressive representation language, with correspondingly sophisticated reasoning capabilities, as is generally required for representing semantic context and the semantic content of TO-acts. People simply do not, for example, use quantifications and modifications when performing turn- or time management acts. Apparent use of modifiers, as in “even kijken” (just a minute), is better interpreted pragmatically, as a way to mitigate the requested break (as also in the common use of the diminutive form in Dutch: “een ogenblikje”, “momentje”, both meaning a little moment), rather than as part of the semantic content. Using a simple parameter-based representation greatly simplifies the task of designing and implementing a context model for effective and efficient interaction management.

It should on the other hand be acknowledged that dialogue participants do form nested and quantified beliefs about IM conditions; for instance, when partner I says “een ogenblikje”, partner C knows that there is some process P, that I intends to execute, that requires substantial time. We can represent this in a formal language with facilities for expressing beliefs by construing the use of parameter specifications as a way to form predicates; using $B_C$ and $B_I$ for belief operators relating to the subscribed partners, we thus want to represent something like:

$$B_C(3P : B_I(\text{TIME-NEEDED} : \text{subst}(P)))$$

On this view a parameter specification like [TIME-NEEDED: subst] is considered as equivalent to $\lambda x : \text{TIME-NEEDED}(x) = \text{subst}$. This amounts to considering a parameter specification as the same thing as a feature specification. In Bunt & van der Sloot (1994) a formal integration of feature structures into a powerful logical representation language has been described, which can be used here to integrate a simple parameter-based representation of IM information into the powerful representation language required for task-oriented semantic information. Such an integration is also needed for dealing with IM acts which are expressed not by using a specific IM-utterance form, but by using a full sentence, as in “I will need some time to find that information in the new time table valid from next Monday”. We have not encountered such cases in our corpus, but they are obviously possible, and their treatment should not be excluded by relying entirely on the simple parameter-based representation.

Parameter- or feature-based representations are attractive not only for their conceptual and computational simplicity, but also because they allow the expression of constraints by means of structure sharing. This seems a useful property in context modelling, since there may be desirable
dependencies between elements in different context dimensions. For instance, when a dialogue has a subdialogue about the topic b, it is also the case that one of the participants has a goal involving b, and one also expects to find utterances with the utterance topic b. The notions 'goal', 'topic', and 'dialogue structure' thus typically all involve the same object; structure sharing may be used to represent this, and thus to exploit for example the interpretation of a topic management act in recognizing the speaker's intention in a subsequent TO-question.

The integration of feature structures defined in Bunt & van der Sloot (1994) supports this way of representing constraints, by introducing a context-dependent recursive, compositional evaluation method of formal-language expressions containing structure sharing markers. This method uses two-place evaluation functions \( V(\phi, C) \), called 'evaluations', which are recursive in their first argument, the expression to be recursive in their first argument, the expression to be evaluated. The second argument, a 'context', is a set of expressions where \( V \) looks for occurrences of a structure sharing marker \([1]\) if \( \phi \) is labelled with that marker. Evaluations are compositional in the following sense. Where a classical valuation function \( F \) is compositional when it has the property

\[
F(\phi) = f(t(F(\phi_1),...,F(\phi_k))) \text{ if } \phi = f(\phi_1,...\phi_k),
\]

i.e., when \( \phi \) is formed from the subexpressions \( \phi_1,...,\phi_k \) by means of the construction \( f \), its value is a function \( f(t) \), defined by \( F \), of the values of the subexpressions; similarly for a covaluation we have:

\[
V(\phi, C) = f(t(V(\phi_1, C_1),...,V(\phi_k, C_k)), \text{ with } C_i \subseteq C \text{ for } i \leq k,
\]

where \( V \) defines not only how the construction \( f \) determines the computation \( f(t) \) to be applied to the values of the subexpressions, but also how the subcontexts \( C_i \) are computed from \( C \) (typically, \( C_i \) is simply \( C \) minus \{\phi\}).

This approach to evaluating expressions with structure sharing is very powerful in that it applies not only within feature structures, but more generally to expressions in a formal language which may contain feature structures as subexpressions. As such, this approach opens the way to deal with constraints between, for instance, elements in a goal in the local semantic context and a parameter specification in the linguistic context.

4 CONCLUSIONS

In this paper we have considered dialogue management issues in two ways. On the one hand, we have looked at the different possible motivations for communicative activity, identifying basically 3 sources: (1) the drive to be rational in achieving one's goals and purposes; (2) the disposition to be cooperative, in acting to help the dialogue partner achieving his goals; and (3) the desire to make communication pleasant, honouring the socially established norms and conventions of interaction. We have seen how these fundamental drives lead to various kinds of task-oriented and dialogue control acts.

On the other hand, we have paid special attention to the subclass of interaction management acts, a subclass of dialogue control acts that play an important role in ensuring the conditions for successful communication. The analysis of the conditions that give rise to such acts, and of the kind of propositional content they may have, has led to the formulation of requirements and suggestions for a structured formal representation of context that may be maintained in a computer dialogue system, and would be essential as a basis for effective, efficient and pleasant communicative behaviour.

REFERENCES


DIALOGUE MANAGEMENT IN A MIXED-INITIATIVE, COOPERATIVE, SPOKEN LANGUAGE SYSTEM

Sandra Williams
BT Laboratories
Martlesham Heath
Ipswich IP5 7RE U.K.
swilliams@info.bt.co.uk.

ABSTRACT

This paper describes a dialogue manager for a spoken language system which allows people to access their email by making a telephone call. The dialogue manager accomplishes three tasks: 1) it controls interactions between the Natural Language Processing (NLP) components; 2) it coordinates inputs from the speech recogniser, requests to and from the database query module and outputs to the text-to-speech module; and 3) it dynamically builds a model of the conversational structure and interprets the dialogue at a higher level than individual utterances. The model of conversation enables mixed initiative dialogues where either the caller or the system can take control. Callers can converse with the system, asking it to search for and read out email messages and to perform a number of common email tasks such as forwarding messages, replying to messages, deleting messages, and filing messages in folders. Anaphoric references and ellipses are resolved by referring to the component of the conversational model which records the dialogue history.

When the dialogue manager cannot find exactly what a user is looking for, it is able to send out a cooperative reply. This reply will give the user some extra information which we hope will be more helpful than just saying “none found”.

1. INTRODUCTION

This paper introduces a dialogue manager for a spoken language system, MailSec, currently under development at BT Labs. MailSec allows people to access their email by making a telephone call [1]. An overview of MailSec is given in part 2. The dialogue manager is part of the NLP component of MailSec which is summarised in part 3.

Automatic access to information by telephone has been available since the mid 1980s. Earlier systems had touch-tone input only, then isolated word speech recognisers were developed that were accurate enough to allow callers to say numbers (digit by digit) and a few keywords ('yes', 'no', and so on). Current commercial systems can handle much larger vocabularies and speaker-independent speech recognisers can cope with a mixture of isolated words, connected digits, and connected alphabetical letters. An example of this is an automated directory enquiry system currently in use at BT Labs [2] which holds a database of several thousand people's names and telephone numbers.

In the future, conversations between humans and machines will be far more natural than they are at present. We foresee that they will become more like human-human conversations but the machine will be a more attentive listener than a human; it will have more consistent reasoning skills than a human; and it will be better and more accurately informed than a human.

MailSec's dialogue manager, described in this paper, goes some of the way towards achieving this. It enables more natural dialogues between the caller and system than has been possible to date. One of the ways MailSec does this is to allow either the caller or MailSec to take the initiative in steering the course of the dialogue (see part 5). MailSec attempts to 'understand' more than individual utterances in isolation. It attempts to 'understand' the entire conversation by building a conversational model (see part 4). This means it is capable of resolving between-turn references (see part 6) and of leaving one request and answering another before returning to the original request. MailSec is able to provide co-operative responses (see part 7) when requested information is not there. This means that it tries to provide some information which might be useful, rather than just saying 'none found'.

2. OVERVIEW OF THE TELEPHONE EMAIL SYSTEM: MAILSEC

MailSec is designed for use by people who are away from their offices and who do not have access to a computer and modem, people who nevertheless want to have access to their email. MailSec enables them to ring up and have their emails listed or read out to them. In addition to this, emails can be forwarded, deleted, replied to, and filed.

The application is similar in functionality to the email part of Sun Microsystems's Speech Acts [3], but our system allows callers to converse in a more natural manner and it incorporates discourse features, such as reference resolution, which SpeechActs does not. Part of a typical conversation between a caller and MailSec might progress as follows:

**Caller:**  List the new messages.
**MailSec:**  You have two new messages, one from Esther Vennell entitled 'Stationery', sent on 20th May and one from Keith Preston entitled 'Visit', sent on 20th May.
**Caller:**  Delete the one from Esther.
**MailSec:**  Delete the message from Esther Vennell entitled 'Stationery'?
**Caller:**  Yes.
**MailSec:**  OK, done.
**Caller:**  Read Keith's email.
**MailSec:**  Message from Keith Preston entitled 'Visit', message reads '.....'.

Spoken Language Systems combine the three technologies of Speech Recognition, Natural Language Processing (NLP) and text-to-speech as shown in Figure 1.

Figure 1 is a schematic diagram to show the technologies involved. It does not reflect the system architecture. For this see [1]. The figure has been simplified and many details such as the telephony interface software, interfaces to various different email servers, and interfaces between the Spoken Language System components themselves have not been shown.

The speech recognition component of MailSec is the BT Labs continuous speech recogniser [4] which allows users to speak in a natural manner without having to leave pauses between words. It is a speaker-independent recogniser which means that it can recognise a wide variety of English speakers regardless of accent or voice pitch and modulation. This has obvious advantages over a speaker-dependent recogniser which would have to be trained on all callers' voices before it would be able to recognise them. The function of the speech recogniser is to recognise the incoming speech, and transcribe it from a speech waveform into a textual representation.

For outgoing speech, MailSec uses the BT Labs text-to-speech (TTS) system, Laureate [5]. Laureate's speech is derived from a real human voice and is one of the most natural-sounding TTS systems currently available. Laureate converts a string of outgoing text into a speech waveform to be played back to the caller.

The NLP part of the system interprets the meaning of the incoming speech from the caller which has been transcribed into text by the speech recogniser. It collects information from an email server, and constructs the replies as text for Laureate to convert to a speech waveform. This paper is concerned with the dialogue manager which controls interactions between the natural language processing (NLP) components; coordinates inputs from the speech recogniser, requests to and from the database query module and outputs to the text-to-speech module whilst dynamically building a model of the conversational structure.
3. OVERVIEW OF THE NLP COMPONENT

The dialogue manager is part of the NLP component of MailSec. Other modules include: a parsing/semantics module, an anaphoric reference and ellipsis resolution module, a database query module, a cooperative response module, and a text generation module. These modules, together with the data structures built by the dialogue manager are shown below in figure 2.

![Diagram](image)

**Figure 2. Schematic diagram of MailSec's NLP component**

Figure 2 is a schematic diagram showing how the dialogue manager relates to the other NLP modules. It leaves out many details of the individual modules and those data sources used by modules other than the dialogue manager and the anaphoric reference and ellipsis resolution module. For a more detailed description of the architecture, see [1].

The dialogue manager controls the flow of processing of data by the other NLP modules and coordinates data flowing from/to the speech recogniser, email server, and TTS. A typical processing sequence proceeds in a clockwise direction in figure 2 from the speech recogniser input, through the parsing/semantics module, then to anaphor/ellipsis resolution, then to database query, then to cooperative response analysis, then to text generation and finally out to text-to-speech. The conversational model is modified and consulted at various stages throughout the processing. Note however that the NLP modules may be called by the dialogue manager in a flexible way. An example of this is when the dialogue manager uses the cooperative response module to modify the query, and then database query is attempted again. Brief descriptions of each module are given below.

3.1 The Dialogue Manager

The dialogue manager maintains the conversation between MailSec and the caller. It coordinates the operation of all other system components and builds a dynamic model of the conversation as it progresses.

Individual utterances cannot be totally meaningful and coherent unless they are interpreted in their correct context within the conversation. If a caller says 'Anna', or 'Yes', or 'Read the next', the semantic representation alone does not give enough information for MailSec to understand and know what to do next. Therefore on receiving the parse for the input utterance, the dialogue manager interprets its meaning in the context of the conversation and builds it in to the conversational model. The model ensures that answers are matched up with corresponding questions, and so on. In this way we build up a representation of the dialogue meaning above the level of individual utterances.

This paper is chiefly concerned with the dialogue manager, its associated data structures, and the functionality it provides.

3.2 The Parsing and Semantics Module

On receiving input from the speech recogniser, the dialogue manager sends it to the parsing and semantics module which interprets the meaning of the caller's utterance. The meaning is represented by what we have called an Extended Logical Form (ELF). The ELF is constructed using the standard PROLOG notation where variables are capitalised, and atoms are lower case. An example of an ELF for 'List any emails from Anna' is:
ELF: list(A),
Expect List: [plural[A]],
Spec List: [message(A), from(A,B), name(B,anna), gender(B,feminine)].

The ELF is a non-recursive form with three parts. The first shows that there exists something, A, that is to be listed. The second is a list of facts indicating what the caller is expecting, the Expect List. In this example, the caller is expecting A to be plural ('emails' in the input string). The Expect List is used by the text generator (together with the database query result and the Spec List) to build an appropriate response. The third part is a specification of what the caller requires ('emails from Anna'). The Expect List and Spec List contain a number of conjoined constraints. In this case, there exists something, A, where A is a message, and A is from B where B is named 'anna' and feminine. The Spec List is used by the dialogue manager as the base for building the database query.

3.3 Conversational Modelling

Our model of the conversation is built dynamically as the conversation progresses and it is based on the theory of Games Structure in Conversation devised by Kowtko et al [6]. The model is our own interpretation of the theory of Games Structure in conversation. Other systems have also successfully made use of this theory: [7] and [8]. The model has two parts:

- The History List which contains all previous utterances. This is used to resolve anaphoric references and ellipsis and for requests from the caller to 'Repeat'.
- The stack of conversational games. This is used by the dialogue manager to track the conversation and 'understand' the significance of an utterance in the context of the dialogue and work out how to respond to the caller.

For more details, see part 4.

3.4 Anaphoric Reference and Ellipsis Resolution Module

This is the module that resolves ellipsis and anaphoric references. Anaphoric references occur when a speaker refers to something mentioned earlier in the dialogue (e.g. 'his' in 'read his email'). Ellipsis occurs when something is missing from a speaker's utterance which can be understood from the context (e.g. 'emails' in 'How many ... from Anna do I have?').

Not all incoming utterances undergo reference resolution. For instance if MailSec has just asked a yes-no question (e.g. 'Do you want to delete the message from Alison Simons entitled "Meeting"') and the caller just answers 'yes' or 'no', then this module is bypassed.

The ellipsis and reference resolution module is described in detail below in part 6.

3.5 Database Query Module

The database query module is part of the system that searches the email database for emails that match the query supplied by the dialogue manager.

3.6 Cooperative Response Module

If the database query does not find anything, the cooperative response module is used to modify the query in order that a cooperative response might be given to the caller. See part 7 for more details.

3.7 Text Generation Module

The text generation module constructs the text for the system's response to the caller.

4. CONVERSATIONAL MODEL

The conversational model consists of two parts: the conversational games stack and the history list. These are described in detail below in 4.1 and 4.2.

4.1 Conversational games stack

The dialogue manager implements the theory of 'Games Structure in Conversation' [6]. According to this theory, conversations can be analysed as a series of 'games' and 'moves', each one denoted by a game beginning (such as the INSTRUCT move: 'Tell me the time.' ) and a game end (such as the ACKNOWLEDGE move: 'OK, Thanks.' ). There can be many, or few, turns in the conversation between a game beginning and a game end. Because of the way games tend to be nested, the overall control structure can be represented as a last-in-first-out push-down stack. This Conversational Games Stack, together with the History List, form the core data structure of the dialogue manager.

Kowtko et al have defined six types of Conversational Games (INSTRUCT, CHECK, QUERY-YN, QUERY-W, EXPLAIN, ALIGN) and 12 kinds of game move. The moves are summarised in tables 1 and 2 below:
| INSTRUCT | Communicates a direct or indirect request or instruction, to be done immediately or shortly. e.g. “You then go down two inches.” |
| CHECK | Checks self-understanding of a previous message or instruction by requesting confirmation directly or indirectly; makes sure that a complicated instruction is understood. e.g. “So you want me to go down two inches?” |
| QUERY-YN, QUERY-W | Yes-No question (QUERY-YN) or open-answer Wh-question (QUERY-W); asks for new or unknown detail about some part of task; does not request clarification about instructions (that would be a CHECK); e.g. “Do you have a cairn?” |
| EXPLAIN | Describes status quo or position in task with respect to the goal; freely offered, not elicited; provides new information. e.g. “I’ve got a cairn.” |
| ALIGN | Checks the other participant’s understanding or accomplishment of a goal; elicits a positive response which closes a larger game; checks alignment of both participants’ plans or positions in task with respect to goal; checks attention, agreement, or readiness. e.g. “Ok?”, meaning, “Are you with me?” |

Table 1: Opening Moves [6]

| CLARIFY | Clarifies or rephrases what has previously been said; usually repeats given or known information; elicited by other person. e.g. “South, two inches.” |
| REPLY-Y, REPLY-N | Affirmative (REPLY-Y) or negative (REPLY-N), elicited response to QUERY-YN, CHECK, or ALIGN; also indicates agreement, disagreement, or denial; e.g. “Yes, I have.” |
| REPLY-W | An elicited reply to QUERY-W or CHECK; can be a response to QUERY-YN that is not easily categorizable as positive or negative (REPLY-Y/N). e.g. “Down” |
| ACKNOWLEDGE | Vocal acknowledgement of having heard and understood; not specifically elicited but often expected before the other speaker will continue; announces readiness to hear next move - in essence a request to ‘please continue’; may close a game. e.g. “All right: or ‘Oh right, I see what you mean.” |
| READY | Indicates intention to begin a new game and focuses attention on oneself, in preparation for the new move; an acknowledgement that the previous game has just been completed, or leaving the previous level or game; consists of a cue-word. e.g. “Now” or “Right.” |

Table 2: Other Moves [6]

The six games correspond with the six kinds of opening move in Table 1. Therefore an Instruct game will always start with an INSTRUCT move, a Check game with a CHECK move, and so on. the READY move is a kind of pre-game signaler, so in effect some games will actually begin with READY.

A game may typically consist of a simple pair of moves, for instance a question (e.g. QUERY-YN) and an answer (e.g. REPLY-N). However, a game may consists of only one move, or it may have more than two moves. For instance, an Explain game may consist of just one EXPLAIN move (e.g. at the beginning of the call, MailSec tells the caller how many new messages there are), but a Query-W game which begins with a QUERY-W, may not end with a REPLY-W, there may be an additional ACKNOWLEDGE move such as “OK” or “Thank you”. In practice, the system ends the game after the REPLY-W move and any spurious ACKNOWLEDGE moves are thrown away.

A typical conversation between MailSec and a caller (recorded at BT Labs) is analysed in Figure 3.
The first talk bubble is the caller speaking: 'Read his message'. This is an INSTRUCT move (an imperative) and it starts an INSTRUCT game. The dialogue manager pushes an INSTRUCT game started by the caller onto the game stack.

The second talk bubble is MailSec's reply: 'I can't work out who you mean ...'. This is a QUERY-W move (a WH question) the start of a QUERY-W game. The dialogue manager pushes a QUERY-W game started by MailSec onto the game stack.

The third bubble is the caller saying 'Peter Wyard'. This is a REPLY-W move and it completes the QUERY-W game. The dialogue manager pops MailSec's QUERY-W game from the game stack.

The final talk bubble is MailSec's REPLY-W move reading out the email and ending the INSTRUCT game. The dialogue manager pops the caller's INSTRUCT game from the game stack.

Note the nested nature of the games. Note also that:

- A turn in the conversation may contain more than one move. This has not been implemented yet.
- A move can end more than one game. An example of this is when a caller is answering a question (e.g. a QUERY-YN) and the speech recogniser does not recognise the input, so MailSec says "Pardon?" (QUERY-W). In repeating the utterance, the caller is both ending the QUERY-YN game as well as the new QUERY-W game.

Our implementation of this theory enables mixed initiative dialogues described in section 5. When the caller's utterance first arrives from the speech recogniser, the dialogue manager builds it into the conversational model. If MailSec has just asked a question, then the dialogue manager attempts first to process the incoming utterance as a reply to the question. If this fails, then it is processed as the start of a new game (usually an INSTRUCT game, a QUERY-W game, or a QUERY-YN game). The output from the parsing and semantics module is used to determine which new game is starting.

If the caller begins a new game unexpectedly, then MailSec's question is retained on the stack so that it can check again with the caller later. For instance, when MailSec asked for the name in the above dialogue, the user could have said 'How many new emails do I have?' This would start a new QUERY-W game on the third level of the stack. MailSec would answer this question immediately, then pop the game off the stack. Seeing two levels already on the stack, it would then ask the user again 'Do you still want to read the message from him?'. If the caller chose to ignore the question again, in our current implementation, MailSec will pop these two levels from the stack and forget them.

The dialogue manager's behaviour can be altered to fit in with the requirements of different applications. For some applications certain pieces of information might be essential and it might be
necessary for the caller to supply them in a particular order. In cases like this, the dialogue manager's behaviour could be altered so that unanswered questions do not disappear from the conversational games stack. There is, of course, a limit to the number of times a caller will tolerate being badgered with the same question time after time, even if the words are varied, before he/she gives up.

4.2 History List

The History List records every utterance generated by the caller, together with the corresponding response by MailSec. The response from MailSec is recorded in the History List in the same entry as the caller's utterance. This may be the result of a database query, or it may be a question MailSec asked the user, e.g. QUERY-W(unknown_sender) for questioning the name of the sender of an email.

The History List is used to resolve ellipsis and anaphoric references (see part 6), and also to respond to user requests to 'Repeat'.

5. MIXED INITIATIVE DIALOGUES

Many existing commercial telephone dialogue systems use prompt and response. The system plays out a prompt and the caller has to speak after the prompt and give the information required. MailSec takes a different approach. We are looking beyond prompt and response and aim at mixed-initiative dialogues where the caller does not necessarily have to answer the system's questions but can choose to ignore them and give some other command or ask the system a question back. Our view of mixed initiative in conversation corresponds with that of Walker and Whittaker [9], but our implementation is not rule-based like theirs.

To demonstrate what we mean, we give examples of these two different types of dialogue below.

First we show an example of a prompt and response dialogue between a caller and the large vocabulary system mentioned in the introduction [2]:

System: Welcome to the BT Corporate Directory Enquiry System.
Caller: Welcome
System: Please say the surname.
Caller: Williams
System: And now please SPELL the surname.
System: Thank you. Please say the first name.
Caller: Sandra
System: Is the name Sandra Williams?
Caller: Yes
System: The number is 01473 605660. I'll repeat that. The number is 01473 605660. If you would like to make another enquiry please hold on, otherwise please replace the handset. Thank you for calling.

This dialogue is controlled by the system which means that the system takes all the initiative in the conversation. The caller cannot say anything other than what has been requested. If the system asks a question, it will insist that the question be answered before the dialogue can progress.

Prompt and response style dialogues are very useful in systems like this that give out very limited and specific information (such as a telephone number) and require very limited and specific information from the caller (e.g. a first name and a surname) in order to do so.

Our system is required to be more flexible than this, and can perform a number of tasks as well as give out information. Since the caller will probably only decide on the next thing to do after receiving some information, it is much better if the caller can ask MailSec directly what he/she wants to do next at the point when he/she decides what this is. The alternative to our flexible dialogue approach would be for the caller to navigate through a long and tedious series of menus before finding the information or task he/she wants. This would break the flow of the conversation, take longer, and interrupt the caller's train of thought. Below is an example of a mixed initiative conversation:
6. REFERENCE AND ELLIPSIS RESOLUTION

Reference and ellipsis are common discourse phenomena in natural dialogues. The reference and ellipsis resolution module is separate from the dialogue manager, but since the resolution process is so dependent on the History List part of the Conversational Model, we include a brief description of our treatment here for completeness, but for more details see [10].

When speaking to MailSec, the caller can refer in a comfortable and natural way to an email mentioned earlier in the conversation. This is illustrated by 'it' in 'Read it' in the following dialogue:

**Caller:** Read the messages from Alison

**MailSec:** You have three messages from Alison Simons. Message from Alison Simons entitled "RE: MailSec", message reads "I agree with Peter's comments about MailSec". Do you want to read the next message?

**Caller:** Read Peter's email about MailSec.

**MailSec:** Message from Peter Wyard entitled "MailSec", message reads "At our last meeting we discussed ...". Do you still want to read the messages from Alison Simons?

The caller asks MailSec to read Alison's messages. MailSec reads the first and asks if the caller wants to read the next (at this point MailSec has control). The caller chooses to ignore this question and asks to read Peter's email that Alison mentioned (now the caller takes control). MailSec reads the message and asks again whether to read messages from Alison Simons. If the caller ignores the question again, then MailSec will not ask again until the caller says 'goodbye', when it will do so as a final reminder. Obviously we can change the behaviour so that MailSec will ask as many times as we program it to, but we do not want the dialogue to become tedious and we want to ensure that the caller is always free to do as he/she likes.

**Caller:** Do I have any messages from Anna?

**MailSec:** You have one new message from Anna Cordon entitled 'MT Meeting'.

**Caller:** Read it.

### 6.1 Reference resolution

Anaphoric references are a specific kind of referring expression and are used when a speaker refers back to something mentioned earlier in the discourse, e.g. "he", "him", "her", "that one", "that way", "the house" can all be anaphoric reference expressions.

**Caller:** Are there any messages from Peter?

**MailSec:** You have two messages from Peter Wyard, one is entitled 'Meeting on Tues' and one is entitled 'BusCat Demo'.

**Caller:** Read his second email.

In the above exchange, the caller’s second utterance contains the anaphoric reference 'his'. The parsing/semantic module represents "Read his second email." as an Extended Logical Form (ELF - see part 3.2):

**ELF:**

```
read(A),
Expect List: [salient(A), singular(A)],
Spec List: [message(A), ord(A,2), named(B,C), gender(B, masculine), from(A,B)]
```

It is the reference resolution process which fills in unknown information in the ELF. It does this by searching back through the preceding logical forms to find a masculine person. We are assuming that the last masculine person mentioned will be the referent of 'his', although this might not always be the case. The ELF output by the parsing and semantics module is thus converted into a Resolved Extended Logical Form (RELF) where named(B,C) is changed to named(B, peter):
6.2 Ellipsis resolution

Ellipsis occurs when something is left out of an utterance which can be determined from what has gone before:

**Caller:** Do I have any messages from Peter?

**MailSec:** You have two messages from Peter Wyard.

**Caller:** Do I have any from David?

The caller has left out the noun 'messages' and has simply said 'any'. For this utterance, the parsing/semantics module will produce the extended logical form:

**ELF:**
\[
A,
\]

**Expect List:** \[
[\]
\]

**Spec List:** \[
[from(A,B),
named(B,david), gender(B,masculine)]
\]

The ellipsis resolution module searches the Spec List for an entity relevant to the domain. In the email domain, entities can be emails, folders, or headers. If there is no entity present, then it searches for the missing information in previous logical forms in the History List. It looks for the most recently mentioned entity, 'messages'. It adds this to the Spec List of the ELF to produce the following RELF:

**RELH:**
\[
list(A),
\]

**Expect List:** \[
[\]
\]

**Spec List:** \[
[from(A,B),
named(B,david), gender(B,masculine)]
\]

7. COOPERATIVE RESPONSES

We are experimenting with cooperative responses for situations when the database query module finds no matching emails. The cooperative response module is separate from the dialogue manager, but it serves as a good example of how the dialogue manager monitors the results from other NLP modules and directs the flow of processing accordingly. If the database query module finds nothing, then rather than just saying 'none found', the dialogue manager attempts to give some information which the user might find helpful. It uses the cooperative response module to generalise the query, without making it so general that any email would match. If some emails match a more general query, then the dialogue manager arranges for a cooperative response to be sent to the user.

**Caller:** Do I have any emails from Anna about elephants?

**MailSec:** No, you have no emails from Anna Cordon about elephants, but you have one message from Keith Preston about elephants.

**Caller:** OK, but do I have any emails at all from Anna?

Here the query has been generalised by dropping the condition 'from Anna' and MailSec has tried to be helpful by looking for other messages about elephants.

In order to generalise the query, the dialogue manager must decide which parts of the user's specification are the most important. Is it more important that the email is from Anna or that the email is about elephants? Or are the two equally important? Here MailSec wrongly assumes that the user is primarily interested in elephants. This whole area of cooperativeness is one which we intend to investigate further in the future.

When the database query is generalised with a successful result, the dialogue manager produced another logical form, the Cooperative Extended Logical Form (CoopELF). For the above example, the RELF and CoopELF are as follows:

**RELH:**
\[
list(A),
\]

**Expect List:** \[
[plural(A)]
\]

**Spec List:** \[
[from(A,B),
named(B,anna), gender(B,feminine),
about(A,elephants)]
\]
8. CONCLUSIONS

We have successfully implemented an experimental dialogue manager for a telephone-based email spoken language system which allows more natural dialogues than existing commercial systems. It enables a mixed initiative dialogue by using a model of 'Games Structure in Conversation' [6]. It resolves between-utterance references and ellipsis. It provides helpful cooperative responses for the caller.

As it develops, the dialogue manager implementation is improving and becoming more robust. In the future it will become more domain independent when we gain experience from porting it to other domains.

In building this dialogue manager we have made a significant step forward towards more natural man-machine dialogues.

ACKNOWLEDGEMENTS

We are grateful to the members of the BT Labs NLP Group, and members of the BT Labs Spoken Language System development team for their help, encouragement and advice during this work. In particular we would like to thank David Franklin, Louise Helliker, Christopher Holt and Alison Simons. Special thanks to Peter Wyard, Edward Kaneen and Stephen Appleby for their work on the parsing and semantics module and to Keith Preston for his work on the database query module.

References


THE DIALOG COMPONENT IN THE WAXHOLM SYSTEM

Rolf Carlson
Department of Speech, Music and Hearing,
Box 70014 KTH,
S-10044 Stockholm, Sweden
rolf@speech.kth.se, www.speech.kth.se

ABSTRACT

In this paper we give an overview of the NLP and dialog component in the Waxholm spoken dialog system. We will discuss how the dialog and the natural language component are modeled from a generic and a domain-specific point of view. Dialog management based on grammar rules and lexical semantic features is implemented in our parser. The notation to describe the syntactic rules has been expanded to cover some of our special needs to model the dialog. The parser is running with two different time scales corresponding to the words in each utterance and to the turns in the dialog. Topic selection is accomplished based on probabilities calculated from user initiatives. Results from parser performance and topic prediction are included in the presentation.

INTRODUCTION

Our research group at KTH has, for some years, been building a generic system in which speech synthesis and speech recognition can be studied in a man-machine dialog framework. The demonstrator application, Waxholm, gives information on boat traffic in the Stockholm archipelago. It references time tables for a fleet of some twenty boats from the Waxholm company which connects about two hundred ports. The system has been presented on several occasions, for example, the Eurospeech '93 conference (Blomberg et al., 1993), the ARPA meeting '94 (Carlson, 1994) and the ETRW on Spoken Dialog Systems (Bertenstam et al., 1995a).

Besides the speech recognition and synthesis components, the system contains modules that handle graphic information such as pictures, maps, charts, and time-tables (Figure 2). This information can be presented to the user at his/her request. The possibility of expanding the task in many directions is an advantage for our future research on interactive dialog systems. In addition to boat timetables, the database also contains information about port locations, hotels, camping places, and restaurants in the Stockholm archipelago. This information is accessed by SQL, the standardized query language. An initial version of the system has been running since September 1992.

The application has similarities to the ATIS domain within the ARPA community, the Voyager system from MIT (Glass et al., 1995) and similar tasks in Europe, for example SUNDIAL (Peckham, 1993), the systems for train timetables information developed by Philips (Aust et al., 1994; Oerder and Aust, 1994) and CSELT (Clementino, and Fissore, 1993; Gerbino and Daniele, 1993; and flight information in the Danish Dialog Project, Dalsgaard and Backgaard, 1994).

Spoken dialog management has attracted considerable interest during the last years. Special workshops and symposia, for example the special workshop at Waseda University, Japan 1993 (Shirai and Furui, 1995), the Twente Workshops on Language Technology in Enschede, The Netherlands, and the 1995 ESCA workshop on Spoken Dialog Systems in Vigo, Denmark, have all been arranged to forward research in this field. We will not attempt to review this growing field in this paper. We will, however, describe in some detail the current effort to model the dialog in the Waxholm system.

Our objective is to develop a dialog management module which can handle the type of interaction that can occur in our chosen domain. The Waxholm system should allow user initiatives, without any specific instructions to the user, complemented by system questions to achieve the user's goal. We will discuss how the dialog and the natural language component are modeled from a generic and a domain-specific point of view.
Figure 1. The graphical model of the WAXHOLM micro-world.

GRAPHICAL INTERFACE

The Waxholm system can be viewed as a micro-world, consisting of harbors with different facilities and with boats that you can take between them. The user gets graphic feedback in the form of tables complemented by speech synthesis. Up to now the subjects have been given a scenario with different numbers of subtasks to solve. A problem with this approach is that the subjects tend to use the same vocabulary as the text in the given scenario. We also observed that the user often did not get enough feedback to be able to decide if the system had the same interpretation of the dialogue as the user. To deal with these problems a graphical representation that visualizes the Waxholm micro-world is being implemented. An example is shown in Figure 1. One purpose of this is to give the subject an idea of what can be done with the system, without expressing it in words. Another purpose is that the interface continuously feeds back the information that the system has obtained from the parsing of the subject's utterance, such as time, departure port and so on. The interface is also meant to give a graphical view of the knowledge the subject has secured thus far, in the form of listings of hotels and so on.

For the speech-output component we have chosen our multi-lingual text-to-speech system (Carlson, Granström and Hunnicutt, 1991). The system is modified for this application. In dialog applications such as the Waxholm system we have a better base for prosodic modeling compared to ordinary text-to-speech, since, in such an environment, we will have access to much more information than if we used an unknown text as input to the speech synthesizer. The speech synthesis has recently been complemented with a face-synthesis module (Beskow, 1995). Both the visual and the speech synthesis are controlled by the same synthesis software.
NATURAL LANGUAGE MODELING

In this section we will give a short review of the natural language component, STINA. Some of our fundamental concepts were initially inspired by TINA, a parser developed at MIT, (Seneff , 1992.) STINA is knowledge based and contains a context-free grammar which is compiled into an ATN. (A detailed description of the parser can be found in Carlson and Hunnicutt, 1995.) Probabilities are assigned to each arc after training. These probabilities are primarily used to reduce search time and for hypothesis pruning.

The parsing is done in three steps. The first step makes use of broad categories such as nouns, while the following step expands these into more detailed solutions. The last step involves recalculation of hypothesis probabilities according to a multi-level N-gram model.

Domain dependent feature system

The feature system used in the parser plays an important role. Each lexical entry can have domain specific semantic features associated to it in addition to the basic syntactic features. The semantic features as used in STINA can be divided into two different classes, basic semantic features and function features. Basic features such as BOAT and PORT give a simple description of the semantic property of a word and are often domain specific. These features are hierarchically structured. In our domain we have specified that a PORT is part of an ISLAND, which is part of a REGION, which is part of a PLACE, which is part of the WORLD.

The second type of semantic features is the "function features." These features are not hierarchical. Typically they are associated with an action, such as TO_PLACE indicating the destination in an utterance regarding travel (Example 1). The function features are also node names in the parser. A verb can have function features set, allowing or disallowing a certain type of modifier to be part of a clause. The action itself in the TO_PLACE example has, of course, a broader scope than the traveling domain, and includes movements between any reference points. Thus, the node TO_PLACE is specified as a prepositional phrase starting with "to" and followed by any nominal expression. The scope of the phrase is changed according to the domain by training.

Example 1:

( TO_PLACE ("to"/TO "Waxholm"/noun) )

The function features are powerful tools to control the analysis of responses to questions from the dialog module. The question "Where do you want to go?" conditions the parser to accept a simple port name or a prepositional phrase.
including a port name as a possible response from the user. This property of STINA gives the parser some of the advantages of a functional grammar parser.

Terminal node evaluation is primarily carried out on the grammatical features. If this basic constraint evaluation is accepted, the semantic features are also evaluated. The hierarchical structure has importance for the rule writing. During the unification process all semantic features which belong to the same semantic branch in the feature tree are considered. The whole tree of the lexical entry is moved into the hypothesis including the leaves on the feature tree. In our traveling domain a port name will keep its PORT feature even if only the PLACE is noted in the grammar. This has several advantages. The rules or terminal specifications do not have to be more specific than necessary and the domain knowledge can, to some extent, be part of the lexicon rather than the rules. This mechanism is extensively used in the sublanguage grammar for our application. In the next section we will see how the introduction of domain dependent terminal nodes is delayed during the parsing process.

**General grammar to subgrammar**

It has been an ambition in our work to create a general grammar which at least covers the type of dialog found in our domain. After a utterance initially has been parsed, we have a hypothesis in terms of grammar nodes and generic terminals such as nouns. In the next step the terminals are replaced by more domain specific labels. A domain specific list of possible terminals is processed by the parser during initialization and each such terminal is associated a generic terminal node. The domain specific nodes are typically constrained by domain specific semantic features in addition to the basic syntactic ones. In our case we have defined a number of terminals, such as port, hotel, boat and time-table. These are all part of the noun class and will replace the "noun" terminal whenever appropriate according to the semantic features of the lexical entry. In our application, then, the lexicon defines that there are nouns with a specific semantic feature, PORT, and is able to separate them from other nouns. In Example 1 the simple phrase is turned into the phrase "TO port" since Waxholm is a port and the terminal port is part of the noun class, Example 2.

**Example 2:**

```
(TO_PLACE ("to"/TO "Waxholm"/port))
```

With this approach we can formulate a general grammar and make it domain specific with the help of the feature system and lexical specifications.

The described method has some attractive side effects. Since the network specified by the ATN has generic terminals, the number of nodes and transitions are less than if the grammar were more specific. This makes the parsing faster since fewer hypotheses have to be evaluated. However, the probability calculation is less informative based on broad categories and has to be reconsidered. In our case this is done with the help of N-gram models.

**N-gram models**

It seems to be a general consensus that N-gram models, in the context of speech understanding, have at least as good predictive power as regular knowledge based grammars (Jelinek et al., 1992, and Stolcke 1995.) However, some research, such as the work by Seneff et al. (1995) has shown, that a knowledge based parser including multilayered probabilities has some advantages. This is specially true for the following processing in the dialog system.

In STINA, smoothed N-gram models are used in addition to the regular transition probabilities. N-gram probabilities are added to the node probabilities after the domain specific node replacements have been performed and before a hypothesis is pushed on the probability ordered N-best stack. The N-gram probabilities include not only terminal node sequences but also phrase level heads. The work by Moore et al. (1995) has earlier shown the advantage in adding phrase heads in the N-gram calculation. In Example 3 the hypothesis score calculation includes for example:

```
p( boat | "TOP+SUBJ"),
p( verb | "SUBJ+boat+VP"),
p( TO | "VP+verb+TO_PLACE"),
p( port | "VP+verb+TO_PLACE").
```

We have expanded the calculation to also include phrase level head node probabilities. However, they are based on phrase level head sequences.

```
p( SUBJ | "TOP"),
p( VP | "TOP+SUBJ"),
p( TO_PLACE | "TOP+SUBJ+VP").
```

**Example 3:**

```
(TOP (SUBJ "båten"/boat) (VP "går"/verb
  (TO_PLACE ("till"/TO "Waxholm"/port))))
```
As an additional example we find that the utterance "I want to go from X to Y" is more probable in our application than "I want to go to X from Y" as reflected in the node N-gram probabilities. Thus, this last step of hypothesis scoring is a powerful method to adjust the general grammar to the domain specific analysis that is needed. Certain phrases and phrase sequences will be well described in the N-gram statistics.

**DIALOG MODELING**

Two major ideas have been guiding the work on the dialog model. First, the dialog should be described by a grammar. Second, the dialog should be probabilistic. In our system, dialog building blocks are described by nodes. Each node has specifications concerning, for example, dialog action, constraint evaluation and system response. A graphical interface to the system presents the dialog grammar graphically. Both the syntax and the dialog can be modeled and edited graphically with this tool.

In the following description, we have used the term “topic” to describe what type of information a user is requesting or, in some cases, a special response from the system. In Table 2, some of the major topics are listed. Topic selection is accomplished based on probabilities calculated from user initiatives (Carlson, 1994; Carlson and Hunnicutt, 1995; Carlson, Hunnicutt and Gustafson, 1995). Lexical semantic information combined with semantic grammar nodes are used as factors in this calculation. The topic selection based on probabilities in our system has similarities with the effort at AT&T (Gorin, 1994; Gorin et al., 1994). A different approach, also based on training, has been presented by Kuhn and De Mori (1994) in their classification approach. A special session in the EuroSpeech 1995 conference was devoted to word spotting including topic spotting based on keywords. The work by Nowell and Moore (1995) goes one step further exploring non-word based topic spotting.

The dialog component is controlled by hand-crafted context free rules which control all steps in the dialog including database search, speech and face synthesis, and other graphical feedback. The dialog component is probabilistic in terms of topic selection based on user initiatives.

Initially the system was designed based on intuition and discussions within the group. After the first subjects were recorded, we were able to base both the grammar and the dialog model on the gathered empirical data. The lexicon was expanded and the network probabilities were trained.

The semantic frame plays an important role in the dialog model. After the syntactic tree has been reduced to a semantic tree, a semantic frame is created with slots corresponding to attribute-value information taken from the tree. The semantic frame has a feature specification describing which features are used in the frame and which

---

**Table 2.** The main topics used in the Waxholm dialog model.
information might have been added to the frame from the dialog history. Each step in the dialog progress is associated to a new frame, with reference pointers to the history. The information that should be pushed forward is defined by semantic feature specifications in the dialog rules.

**Dialog rules**

Each predicted dialog topic is explored according a set of rules. These rules define which constraints have to be fulfilled and what action should be taken depending on the dialog history. A node can be a terminal node or the entry point to a network. In the case of a terminal, several node specific characteristics have to be defined in the rule system. These can divided into three basic groups: constraint evaluation, system questions and system actions.

The constraint evaluation is described in terms of features and in terms of the content in the semantic frame. If the frame needs to be expanded with additional information, a system question is synthesized. During recognition of a response to such a question, the grammar is controlled with semantic features in order to allow incomplete sentences. This whole process is handled by the feature passing between the dialog part and the grammar part of STINA.

If the response from the subject does not clarify the question, the robust parsing is temporarily disconnected so that informative error messages can be given to the user about lexical or syntactic problems. At the same time, a complete sentence is requested giving the dialog manager the possibility of evaluating whether the chosen topic is incorrect. This technique has been shown to be useful in helping subjects to recover from an error through rephrasing of their last input (Hunnicutt et al., 1992).

In most cases, the user does answer system questions, (Bertenstam et al., 1995b, 1995c). A positive response from the constraint evaluation opens the way for the selected action to take place, such as a database search or a graphic presentation of a map or a table.

Most of the terminal nodes have no direct input from the user. Rather they deal with small details such as making the synthetic face to look at a graphic object on the screen or to simply synthesize a message to the user. Thus, the dialog rules do not only model turns in an oral dialog, they cover all actions in the dialog system. The expanded grammar notation makes it possible to separate the dialog model from the system implementation.

A modification of the domain implies an addition of how to handle a new topic, but it is our ambition that the implementation and the training procedures should, as much as possible, be kept the same.

**Topic selection**

The decision about which topic path to follow in the dialog is based on several factors such as the dialog history and the content of the specific utterance. The utterance is coded in the form of a "semantic frame" with slots corresponding to both the grammatical analysis and the specific application. The structure of the semantic frame is automatically created based on the rule system.

Each semantic feature found in the syntactic and semantic analysis is considered in the form of a conditional probability to decide on the topic. The probability for each topic is expressed as: \( p(\text{topic}|F) \), where \( F \) is a feature vector including all semantic features used in the utterance. Thus, the \( \text{BOAT} \) feature can be a strong indication for the \$\text{TIME\_TABLE}\$ topic but this can be contradicted by a \$\text{HOTEL}\$ feature. The topic prediction has been trained using a labeled set of utterances taken from the Waxholm database. Only utterances indicating a topic (about 1200) have been included in this set. The probability is calculated according to: \( p = \frac{(n+1)(N+2)}{N(n+1)(N+2)} \), where \( N \) = number of times a feature can be a terminal node in the feature tree, and \( n \) = number of times a feature actually is a terminal node in a topic indicating utterance.

**Introduction of a new topic**

In this section we will give a simple example of how a new topic can be introduced. Suppose we want to create a topic called "out of domain." First a topic node is introduced in the rule system. Some new words probably need to be included in the lexicon and labeled with a semantic feature showing that the system does not know how to deal with the subjects these words relate to. Then a synthesis node might be added with an output informing the user about the situation such as "We are not able to process things dealing with booking". Example sentences must be created that illustrate the problem and the dialog parser must be trained with these sentences labeled with the "out of domain" topic. Since the topic selection is done by a probabilistic approach that needs application-specific training, data collection is of great importance.
MAN-STINA INTERACTION

In the implementation of the parser and the dialog management, we have stressed an interactive development environment. It is possible to study the parsing and the dialog flow step by step when a graphic tree is built. It is even possible to use log files collected during Wizard of Oz experiments as scripts to repeat a specific dialog, including all graphic displays and acoustic outputs.

We have added a graphical interface to the system which presents each network graphically. Both the syntax and the dialog networks can be modeled and edited graphically with this tool. Earlier work on dialog modeling such as the Generic Dialog System Platform in the Danish dialog project (Larsen and Backgaard, 1994) has been an inspiration for this expansion. This platform is based on a special tool in which the dialog flow can be described by a network of building blocks. These blocks can be edited graphically. The OASIS developed by the GTE Laboratories Incorporated (Zeigler and Mazor, 1994) should also be referred to in this context. It is based on dialog prototypes which include building blocks for the acquisition of factual information, for the verification of acquired information, and for reacquisition following a disconfirmation.

EVALUATION OF THE NLP AND DIALOG MODULES

Evaluation of the system has been performed using part of the Waxholm database. In this database, speech and text data was collected using the Waxholm system. Initially, a “Wizard of Oz” replaced the speech recognition module. A full report on the data collection and data analysis can be found in Bertenstam et al. (1995b and 1995c.)

The database was collected using preliminary versions of each module in the Waxholm system. This procedure has advantages and disadvantages for the contents of the database. System limitations will already from the beginning put constraints on the dialog, making it representative for a human-machine interaction. However, since the system was under development during the data collection, it was influenced by the system status at each recording time. After about half of the recording sessions, the system was reasonably stable, and the number of system “misunderstandings” had been reduced. As research on dialog systems develops, it becomes more important to develop new methods to evaluate human-machine interaction (Hirschman and Pao, 1993.)

Test material

The test material used in the experiments includes 68 subjects and 1900 dialog utterances containing 9200 words. The total recording time amounts to 2 hours and 16 minutes. The most frequent 200 words out of the total of 720 words cover 92 percent of the collected transcribed data. About 700 utterances are simple answers to system questions while the rest, 1200, can be regarded as user initiatives.

We can find a few examples of restarts in the database due to hesitations or mistakes on the semantic, grammatical or phonetic level. However, less than 3% of the utterances contain such disfluencies. Some of the restarts are exact repetitions of a word or a phrase. In some cases a preposition, a question word or a content word is changed. The average utterance length was 5.6 words. The average length of the first sentence in each scenario was 8.8 words.

Parser evaluation

The parser has been evaluated in several different ways. Most tests used a deleted estimation procedure. Using about 1700 sentences in the Waxholm database, 62 percent give a complete parse, whereas if we restrict the data to utterances containing user initiatives (about 1200), the result is reduced to 48 percent. This can be explained by the fact that the large number of responses to system questions typically have a very simple syntax.

If we exclude extralinguistic sounds such as lip smack, sigh and laughing in the test material based on dialog initiatives by the user, we increase the result to 60 percent complete parses. Sentences with incomplete parses are handled by the robust parsing component and frequently effect the desired system response.

The perplexity on the Waxholm material is about 26 using a trained grammar. If only utterances with complete parses are considered we get a perplexity of 23.

N-best resorting

The parser has also been evaluated in an N-best list resorting framework. Totally 290 N-best lists with about 10 alternatives each were generated, using an early version of the speech recognition module of the Waxholm system (Ström, 1995.) Since several of the utterances were answers to simple questions the utterance length only averaged about 5 words. The top choice using a bigram
grammar as part of the recognition module gave a word accuracy of 76.0%. The mean worst and best possible accuracy in the lists were 48.0% and 86.1%. After resorting using the STINA parser the result improved to 78.6% corresponding to about 25% of the possible increase.

<table>
<thead>
<tr>
<th>Test material</th>
<th>N</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>woz input</td>
<td>1209</td>
<td>12.9</td>
</tr>
<tr>
<td>no extralinguistic sounds</td>
<td>1214</td>
<td>12.7</td>
</tr>
<tr>
<td>only complete parses</td>
<td>581</td>
<td>3.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>All topics excluding no “understanding”</th>
<th>N</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>woz input</td>
<td>1154</td>
<td>8.8</td>
</tr>
<tr>
<td>no extralinguistic sounds</td>
<td>1159</td>
<td>8.5</td>
</tr>
<tr>
<td>only complete parses</td>
<td>580</td>
<td>2.9</td>
</tr>
</tbody>
</table>

Table 3. Results from the topic prediction experiments.

**Evaluation of topic selection**

We have performed a sequence of tests to evaluate the topic selection method. The evaluation has used one quarter of the material, about 300 utterances, as test material, and the rest as training material, about 900 utterances. This procedure has been repeated for all quarters and the reported results are the mean values from these four runs. The first result, 12.9% errors in Table 3, is based on the unprocessed labeled input transcription. The eight possible topics have a rather uneven distribution in the material with TIME_TABLE occurring 45% of the time. One of the topics, labeled “no understanding,” is trained on a set of constructed utterances that are not possible to understand, even for a human. This topic is then used as a model for the system to give an appropriate “no understanding” system response. It should be noted that, in principle, these utterances still can have a reasonable parse. However, the topic prediction is certainly influenced by a poor parse. It seemed reasonable to exclude the “no understanding” prediction from the result since the system at least does not make an erroneous decision.

The accuracy model in word recognition evaluation has the same underlying principle. By excluding 55 utterances, about 5% of the test corpus, predicted to be part of the “no understanding” topic, we reduce the error by about 4%.

In the next experiment, we excluded all extralinguistic sounds, about 700, in the input text. This will increase the number of complete parses with about 10% as discussed earlier. The prediction result was about the same compared to the first experiment.

The final experiment included only those utterances that gave a complete parse in the analysis. The errors were drastically reduced. We do not yet know if an increased grammatical coverage also will reduce the topic prediction errors.

**SUMMARY**

Lexical semantic information combined with the grammar rules describe the system constraints in our system. Thus, the choice of semantic features and terminal nodes will automatically turn the general grammar into a subgrammar based on the domain. The use of N-gram statistics improves the predictive power of the grammar on both terminal level and phrase structure. Topic prediction based on semantic features separates the surface form of an utterance from the intention of the subject. The dialog design can be data driven to some extent with the proposed method. The rule-based, and to some extent, probabilistic approach we are exploring makes the addition of new topics relatively easy. However, much manual work still remains to be done when an application domain should be changed.

**ACKNOWLEDGMENT**

We thank all the people in the Waxholm project group who have been engaged in data collection, data analysis and system design. The Waxholm group consists of staff and students at the Department of Speech, Music and Hearing, KTH. Most of the efforts are done part time. The members of the group in alphabetic order are: Johan Berstenam, Jonas Beskow, Mats Blomberg, Rolf Carlson, Kjell Elénius, Björn Granström, Joakim Gustafson, Sheri Hunnicutt, Jesper Högborg, Roger Lindell, Lennart Neovius, Lennart Nord, Antonio de Serpa-Leitão and Nikko Ström. This work has been supported by grants from The Swedish National Language Technology Program.
REFERENCES


PROJECT PARLEVINK

Language Engineering
University of Twente

Parlevink
The Parlevink project is a language theory and technology project of the University of Twente. Starting-point is a (software) engineering approach to natural language and natural language processing systems. Special attention is paid to possible interactions with theoretical computer science.

Research Topics
In subprojects research is devoted to topics as syntax, semantics and pragmatics. Dialogue modeling is also part of the project, just as connectionist language learning and processing. Support research is done in the areas of formal languages and neural networks. Integration of the different topics takes place in the SCHISMA subproject: design and realization of a prototype natural language accessible theater information and booking system (joint with KPN Research). Other topics that play a role in this integration project are the embedding of such a system in a 'digital city' and World Wide Web, speech processing, natural language technology assessment and societal aspects. Another integration project is the European funded 21-project on multimedia information retrieval. Other projects that will start in 1996 are on neural networks and on societal effects of scientific research.

Ph.D. Research
In 1996 four to five Ph.D. students will perform their research in this project, partly in cooperation with other projects (robust language analysis, pragmatics, dialogue modeling and analysis, design and specification of NLP systems). In the spring of 1995 the Ph.D. thesis ‘Little Linguistic Creatures’ appeared. In 1996 a Ph.D. thesis on pragmatics in language technology is expected.

Students
It is expected that in the forthcoming years about thirty students will do their M.Sc. research in the Parlevink project. In addition there are many students who take a practical term outside the university (language and neuro engineering).

Workshops
Twice a year a Twente Workshop on Language Technology (TWLT) is organized. Starting as a local event these workshops have now become meetings of international specialists on language engineering topics. In 1996 two workshops will be organized. The first one, in June, will be on "Dialogue Modeling in Natural Language Systems" (TWLT11); a selection of the papers will appear in a book published by John Benjamins Publishing Company (New York). The second one, in September will be on "Computational Humor: Automatic Interpretation and Generation of Verbal Humor" (IWCH'96/TWLT12).
Twente Workshops
on Language Technology

The TWLT workshops are organised by the PARLEVINK project of the University of Twente. The first workshop was held in Enschede, the Netherlands on March 22, 1991. The workshop was attended by about 40 participants. The contents of the proceedings are given below.

Proceedings Twente Workshop on Language Technology 1 (TWLT 1)
Tomita's Algorithm: Extensions and Applications
Eds. R. Heemels, A. Nijholt & K. Sikkel, 103 pages.

Preface and Contents
A. Nijholt (University of Twente, Enschede). (Generalised) LR Parsing: From Knuth to Tomita.
G.J. van der Steen (Vleermuis Software Research, Utrecht). Unrestricted On-Line Parsing and Transduction with Graph Structured Stacks.
T. Vosse (NICI, Nijmegen). Detection and Correction of Morpho-Syntactic Errors in Shift-Reduce Parsing.
R. Heemels (Océ Nederland, Venlo). Tomita's Algorithm in Practical Applications.
M. Lankhorst (University of Twente, Enschede). An Empirical Comparison of Generalised LR Tables.
K. Sikkel (University of Twente, Enschede). Bottom-Up Parallelization of Tomita's Algorithm.

The second workshop in the series (TWLT 2) has been held on November 20, 1991. The workshop was attended by more than 70 researchers from industry and university. The contents of the proceedings are given below.

Proceedings Twente Workshop on Language Technology 2 (TWLT 2)
Linguistic Engineering: Tools and Products

Preface and Contents
A. Nijholt (University of Twente, Enschede). Linguistic Engineering: A Survey.
B. van Bakel (University of Nijmegen, Nijmegen). Semantic Analysis of Chemical Texts.
T. Vosse (NICI, Nijmegen). Detecting and Correcting Morpho-syntactic Errors in Real Texts.
A. van Rijn (CIAD/Delft University of Technology, Delft). A Natural Language Interface for a Flexible Assembly Cell.
J. Honig (Delft University of Technology, Delft). Using Delta in Natural Language Front-ends.
D. van den Akker (IBM Research, Amsterdam). Language Technology at IBM Nederland.
The third workshop in the series (TWLT 3) was held on May 12 and 13, 1992. Contrary to the previous workshops it had an international character with eighty participants from the U.S.A., India, Great Britain, Ireland, Italy, Germany, France, Belgium and the Netherlands. The proceedings were available at the workshop. The contents of the proceedings are given below.

Proceedings Twente Workshop on Language Technology 3 (TWLT 3)

Connectionism and Natural Language Processing
Eds. M.F.J. Drossaers & A. Nijholt, 142 pages.

Preface and Contents

L.P.J. Veelenturf (University of Twente, Enschede). Representation of Spoken Words in a Self-Organising Neural Net.

P. Wittenburg & U. H. Frauenfelder (Max-Planck Institute, Nijmegen). Modelling the Human Mental Lexicon with Self-Organising Feature Maps.


W. Daelemans & A. van den Bosch (Tilburg University, Tilburg). Generalisation Performance of Back Propagation Learning on a Syllabification Task.

E.-J. van der Linden & W. Kraaij (Tilburg University, Tilburg). Representation of Idioms in Connectionist Models.

J.C. Scholtes (University of Amsterdam, Amsterdam). Neural Data Oriented Parsing.


M.F.J. Drossaers (University of Twente, Enschede). Hopfield Models as Neural-Network Acceptors.


R. Reilly (University College, Dublin). An Exploration of Clause Boundary Effects in SRN Representations.

S.M. Lucas (University of Essex, Colchester). Syntactic Neural Networks for Natural Language Processing.

R. Milkkulainen (University of Texas, Austin). DISCERN: A Distributed Neural Network Model of Script Processing and Memory.

The fourth workshop in the series has been held on September 23, 1992. The theme of this workshop was "Pragmatics in Language Technology". Its aim was to bring together the several approaches to this subject: philosophical, linguistic and logic. The workshop was visited by more than 50 researchers in these fields, together with several computer scientists. The contents of the proceedings are given below.

Proceedings Twente Workshop on Language Technology 4 (TWLT 4)

Pragmatics in Language Technology

Preface and Contents

D. Nauta, A. Nijholt & J. Schaake (University of Twente, Enschede). Pragmatics in Language technology: Introduction.

Part 1: Pragmatics and Semiotics

J. van der Lubbe & D. Nauta (Delft University of Technology & University of Twente, Enschede). Semiotics, Pragmatism, and Expert Systems.

F. Vandamme (Ghent). Semiotics, Epistemology, and Human Action.

H. de Jong & W. Werner (University of Twente, Enschede). Separation of Powers and Semiotic Processes.
Part 2: Functional Approach in Linguistics
C. de Groot (University of Amsterdam). Pragmatics in Functional Grammar.
E. Steiner (University of Saarland, Saarbrücken). Systemic Functional Grammar.
R. Bartsch (University of Amsterdam). Concept Formation on the Basis of Utterances in Situations.
Part 3: Logic of Belief, Utterance, and Intention
J. Schaaeke (University of Twente, Enschede). The Logic of Peirce's Existential Graphs.
H. Bunt (Tilburg University). Belief Contexts in Human-Computer Dialogue.

The fifth workshop in the series took place on 3 and 4 June 1993. It was devoted to the topic "Natural Language Interfaces". The aim was to provide an international platform for commerce, technology and science to present the advances and current state of the art in this area of research.

Proceedings Twente Workshop on Language Technology 5 (TWLT 5)
Natural Language Interfaces
Eds. F.M.G. de Jong & A. Nijholt, 124 pages.

Preface and Contents
F.M.G. de Jong & A. Nijholt (University of Twente). Natural Language Interfaces: Introduction.
L. Boves (University of Nijmegen). Spoken Language Interfaces.
J. Nerbonne (University of Groningen). NL Interfaces and the Turing Test.
D. Speelman (University of Leuven). A Natural Language Interface that Uses Generalised Quantifiers.
W. Menzel (University of Hamburg). Title.
G. Neumann (University of Saarbrücken). Design Principles of the DISCO system.

The sixth workshop in the series took place on 16 and 17 December 1993. It was devoted to the topic "Natural Language Parsing". The aim was to provide an international platform for technology and science to present the advances and current state of the art in this area of research, in particular research that aims at analysing real-world text and real-world speech and keyboard input.

Proceedings Twente Workshop on Language Technology 6 (TWLT 6)
Natural Language Parsing: Methods and Formalisms
Eds. K. Sikkel & A. Nijholt, 190 pages.

Preface and Contents
A. Nijholt (University of Twente). Natural Language Parsing: An Introduction.
V. Manca (University of Pisa). Typology and Logical Structure of Natural Languages.
R. Bod (University of Amsterdam). Data Oriented Parsing as a General Framework for Stochastic Language Processing.

M. Stefanova & W. ter Stal (University of Sofia / University of Twente). A Comparison of ALE and PATR: Practical Experiences.

J.P.M. de Vreught (University of Delft). A Practical Comparison between Parallel Tabular Recognizers.

M. Verlinden (University of Twente). Head-Corner Parsing of Unification Grammars: A Case Study.


Th. Stürmer (University of Saarbrücken). Semantic-Oriented Chart Parsing with Defaults.

G. Satta (University of Venice). The Parsing Problem for Tree-Adjoining Grammars.

F. Barthélémy (University of Lisbon). A Single Formalism for a Wide Range of Parsers for DCGs.


C. Cremers (University of Leiden). Coordination as a Parsing Problem.

M. Wirén (University of Saarbrücken). Bounded Incremental Parsing.

V. Kubon and M. Plátek (Charles University, Prague). Robust Parsing and Grammar Checking of Free Word Order Languages.

V. Srinivasan (University of Mainz). Punctuation and Parsing of Real-World Texts.

T.G. Vosse (University of Leiden). Robust GLR Parsing for Grammar-Based Spelling Correction.

The seventh workshop in the series took place on 15 and 16 June 1994. It was devoted to the topic "Computer-Assisted Language Learning" (CALL). The aim was to present both the state of the art in CALL and the new perspectives in the research and development of software that is meant to be used in a language curriculum. By the mix of themes addressed in the papers and demonstrations, we hoped to bring about the exchange of ideas between people of various backgrounds.

Proceedings Twente Workshop on Language Technology 7 (TWLT 7)

Computer-Assisted Language Learning

Eds. L. Appelo, F.M.G. de Jong, 133 pages.

Preface and Contents


M. van Bodegom (Eurolinguist Language House, Nijmegen, The Netherlands). Eurolinguist test: An adaptive testing system.

B. Cartigny (Escape, Tilburg, The Netherlands). Discatex CD-ROM XA.

H. Altay Guvenir, K. Offazer (Bilken University, Ankara). Using a Corpus for Teaching Turkish Morphology.


G. Kempen, A. Dijkstra (University of Leiden, The Netherlands). Towards an integrated system for spelling, grammar and writing instruction.

F. Kronenberg, A. Krueger, P. Ludewig (University of Osnabrueck, Germany). Contextual vocabulary learning with CAVOL.

S. Lobbe (Rotterdam Polytechnic Informatica Centrum, The Netherlands). Teachers, Students and IT: how to get teachers to integrate IT into the (language) curriculum.


B. Suverda (SLO, Enschede, The Netherlands). Developing a Multimedia Course for Learning Dutch as a Second Language.
C. Schwind (*Universite de Marseille, France*). Error analysis and explanation in knowledge based language tutoring.

J. Thompson (*CTI, Hull, United Kingdom/EUROCALL*). TELL into the mainstream curriculum.

M. Zöck (*Limsi, Paris, France*). Language in action, or learning a language by watching how it works.

Description of systems demonstrated:

- **APPEAL** (*Institute of Perception Research, Eindhoven*)
- **Bonacord, Mél-Mélo, etc.* (*School of European Languages & Cultures, University of Hull*)
- **Computer BBS in language instruction** (*English Programs for Internationals, University of South Carolina*)
- **Discetext** (*Escape, Tilburg*)
- **Error analysis and explanation** (*CNRS, Laboratoire d'Informatique de Marseille*)
- **ItalCultura, RumboHispano and IVANA** (*Norwegian Computing Centre for the Humanities, Harald*)
- **It's English** (*Department of Educational Sciences, Utrecht University*)
- **Multimedia course for learning Dutch** (*SLO, Enschede*)
- **Part of CATT** (*Department of Computer Engineering and Information Science, Bilkent University, Ankara*)
- **PROMISE** (*Institut für Semantische Informationsverarbeitung, Universität Osnabrück*)
- **Speech-Melody trainer** (*Institute of Perception Research, Eindhoven*)
- **The Rosetta Stone** (*Eurolinguist Language House Nijmegen*)
- **Verbarium and Substantarium** (*SOS Nijmegen*)
- **WOORD** (*Applied Linguistics Unit, Delft University of Technology*)
- **FLUENT-II** (*George Mason University, Washington*)

The eighth workshop in the series took place on 1 and 2 December 1994. It was devoted to speech, the integration of speech and natural language processing, and the application of this integration in natural language interfaces. The program emphasized research of interest for the themes in the framework of the Dutch NWO programme on Speech and Natural Language that started in 1994.

**Proceedings Twente Workshop on Language Technology 8 (TWLT 8)**

*Speech and Language Engineering*

Eds. L. Boves, A. Nijholt, 176 pages.

Preface and Contents

- **Chr. Dugast (Phillips, Aachen, Germany)**. The North American Business News Task: Speaker Independent, Unlimited Vocabulary Article Dictation
- **P. van Alphen, C. In't Veld & W. Schelvis** (*PTT Research, Leidschendam, The Netherlands*). Analysis of the Dutch Polyphone Corpus.
- **J.M. McQueen** (*Max Planck Institute, Nijmegen, The Netherlands*). The Role of Prosody in Human Speech Recognition.
- **L. ten Bosch** (*IPO, Eindhoven, the Netherlands*). The Potential Role of Prosody in Automatic Speech Recognition.
- **M.F.J. Drossaers & D. Dokter** (*University of Twente, Enschede, the Netherlands*). Simple Speech Recognition with Little Linguistic Creatures.
- **H. Helbig & A. Mertens** (*FernUniversität Hagen, Germany*). Word Agent Based Natural Language Processing.
- **Geunbae Lee et al.** (*Pohang University, Hyoja-Dong, Pohang, Korea*). Phoneme-Level Speech and natural Language Integration for Agglutinative Languages.
The ninth workshop in the series took place on 9 June 1995. It was devoted to empirical methods in the analysis of dialogues, and the use of corpora of dialogues in building dialogue systems. The aim was to discuss the methods of corpus analysis, as well as results of corpus analysis and the application of such results.

Proceedings Twente Workshop on Language Technology 9 (TWLT 9)  
Corpus-based Approaches to Dialogue Modelling  

Preface and Contents
N. Dahlbäck (NLP Laboratory, Linköping, Sweden). Kinds of agents and types of dialogues.
J.H. Connolly, A.A. Clarke, S.W. Garner & H.K. Palmén (Loughborough University of Technology, UK).  
Clause-internal structure in spoken dialogue.
The coding of dialogue structure in a corpus.
J. Alexanderson & N. Reithinger (DFKI, Saarbrücken, Germany). Designing the dialogue component in a  
speech translation system – a corpus-based approach.
M. Rats (ITK, Tilburg, the Netherlands). Referring to topics – a corpus-based study.
H. Dybkjær, L. Dybkjær & N.O. Bernsen (Centre for Cognitive Science, Roskilde, Denmark). Design,  
formalization and evaluation of spoken language dialogue.
D.G. Novick & B. Hansen (Oregon Graduate Institute of Science and Technology, Portland, USA). Mutuality  
strategies for reference in task-oriented dialogue.
N. Fraser (Vocalis Ltd, Cambridge, UK). Messy data, what can we learn from it?
J.A. Andermarch (University of Twente, Enschede, the Netherlands). Predicting and interpreting speech acts in  
a theatre information and booking system.

The tenth workshop in the series took place on 6-8 December 1995. This workshop was  
organized in the framework provided by the Algebraic Methodology and Software  
Technology movement (AMAST). It focussed on algebraic methods in formal languages,  
programming languages and natural languages. Its aim was to bring together those researchers  
on formal language theory, programming language theory and natural language description  
theory, that have a common interest in the use of algebraic methods to describe syntactic,  
semantic and pragmatic properties of language.
Proceedings Twente Workshop on Language Technology 10 (TWLT 10)
Algebraic Methods in Language Processing
Eds. A. Nijholt, G. Scollo and R. Steetskamp, 263 pages

Preface and Contents
Teodor Rus (Iowa City, USA). Algebraic Processing of Programming Languages.
Eelco Visser (Amsterdam, NL). Polymorphic Syntax Definition.
Teodor Rus & James, S. Jones (Iowa City, USA). Multi-layered Pipeline Parsing from Multi-axiom Grammars.
Michael Moortgat (Utrecht, NL). Multimodal Linguistic Inference.
Annies V. Groenink (Amsterdam, NL). A Simple Uniform Semantics for Concatenation-Based Grammar.
Grzegorz Rozenberg (Leiden, NL). Theory of Texts (abstract only).
Jan Rekers (Leiden, NL) & A Schürr (Aachen, D). A Graph Grammar Approach to Graphical Parsing.
Sándor Horváth (Debrecen, H). Strong Interchangeability and Nonlinearity of Primitive Words.
Theo M.V. Jansen (Amsterdam, NL). The Method of ROSETTA, Natural Language Translation Using Algebras.
Pál Dömös (Kussuth University, H) & Jürgen Duske (University of Hanover, G). Subword Membership Problem for Linear Indexed Languages.
Vincenzo Manca (Pisa, I). A Logical Formalism for Intergrammatical Representation.

The eleventh workshop in the series took place on 19-21 June 1996. It focussed on the task of dialogue management in natural-language processing systems. The aim was to discuss advances in dialogue management strategies and design methods. During the workshop, there was a separate session concerned with evaluation methods.

Proceedings Twente Workshop on Language Technology 11 (TWLT 11)
Dialogue Management in Natural Language Systems
Eds. S. LuperFoy, A. Nijholt and G. Veldhuijzen van Zanten, 228 pages

Preface and Contents
Pierre Nugues, Christophe Godéreaux, Pierre-Olivier and Frédéric Revolta (GREYC, F). A Conversational Agent to Navigate in Virtual Worlds.
Anne Vilnat (LIMSI-CNRS, F). Which Processes to Manage Human-Machine Dialogue?
Latifa Taleb (INRIA, F). Communication Deviation in Finalized InformativeDialogue Management.
Joris Hulstijn, René Steetskamp, Hugo ter Doest (University of Twente, NL), Stan van de Burgt (KPN Research, NL) and Anton Nijholt (University of Twente, NL). Topics in SCHISMA Dialogues.
Gavin Churcher, Clive Souter and Eric S. Atwell (Leeds University, UK). Dialogues in Air Traffic Control
Elisabeth Maier (DFKI, D). Context Construction as Subtask of Dialogue Processing -- the VERBMOBIL Case.
Wieland Eckert (University of Erlangen, D). Understanding of Spontaneous Utterances in Human-Machine-Dialog.
Jennifer Chu-Carroll (University of Delaware, USA). Response Generation in Collaborative Dialogue Interactions.
Harry Bunt (Tilburg University, NL). Interaction Management Functions and Context Representation Requirements.
Peter Wyard and Sandra Williams (BT, GB). Dialogue Management in a Mixed-Initiative, Cooperative, Spoken Language System.
Rolf Carlson (KTH, SW). The Dialog Component in the Waxholm System.

The proceedings of the workshops can be ordered from Vakgroep SETI, Department of Computer Science, University of Twente, P.O. Box 217, NL-7500 AE Enschede, The Netherlands. E-mail orders are possible: twi_secrlcs.utwente.nl. Each of the proceedings costs Dfl. 50.
EVALUATION OF SPOKEN DIALOGUE SYSTEMS

Laila Dybkjaer, Niels Ole Bernsen and Hans Dybkjaer
Centre for Cognitive Science, Roskilde University
PO Box 260, DK-4000 Roskilde, Denmark
emails: laila@cog.ruc.dk, nob@cog.ruc.dk, dybkjaer@cog.ruc.dk
phone: +45 46 75 77 11  fax: +45 46 75 45 02  url: http://www.cog.ruc.dk/

ABSTRACT
As spoken language dialogue systems (SLDSs) are taking off commercially, strong needs are being felt for improved methods and tools to support the evaluation of SLDS designs and products. Little is still known on dialogue evaluation and much work remains to be done. Based on development and evaluation of the dialogue component of an advanced SLDS, the paper reviews the evaluation procedures used and suggests improvements for use in future development projects. Concepts, methods and tools are described, results presented, and improvements proposed.

1. INTRODUCTION
The commercialisation of integrated spoken language dialogue systems (SLDSs) is a contemporary fact. Within the last few years SLDSs have matured to the point of attracting broad industrial interest and commercial SLDSs are now able to carry out routine tasks that were previously done by humans, thus generating significant savings in the companies or public institutions that install them. One of the most advanced systems currently in public use in Europe was introduced in 1994 by the Swedish Telecom Telia to automate part of the directory enquiries task [Forssten 1994].

Along with this development strong needs have arisen for effective evaluation procedures to be used during and after the development of SLDS products. In consequence, speech and natural language systems evaluation is emerging as a scientific sub-discipline in its own right. [Hirschman and Thompson 1996] Work on SLDSs evaluation has received significant stimulation from the ARPA Spoken Language Technology initiative [Galliers and Jones 1993, ARPA 1994] and progress is being made in Europe as well [Eagles 1995]. Progress and established methods exist for the objective evaluation of some of the individual components that make up SLDSs, such as speech recognition and speech synthesis, and objective evaluation procedures are beginning to appear for natural language parsing [Black 1996]. Still, evaluation of SLDSs today remains as much of an art and a craft as it is an exact science with established standards and procedures of good engineering practice. In particular, little is still known on dialogue evaluation including evaluation of dialogue components and integrated SLDSs. Thus,

- little is known about diagnostic evaluation [Hirschman and Thompson 1996], i.e. detection and diagnosis of errors, of dialogue components apart from traditional glass box and black box evaluation;
- little is known about systematic performance evaluation of dialogue components [Hirschman and Thompson 1996], i.e. measurements of the performance of the system in terms of a set of quantitative parameters;
- little is known about adequacy evaluation of integrated SLDSs [Hirschman and Thompson 1996], i.e., about how well a particular SLDS fits its purpose and meets actual user needs and expectations.

As SLDSs are being brought to the market, customer satisfaction becomes an important competitive parameter and hence an important element in measuring the success of an SLDS. However, user satisfaction does not necessarily derive from high technical performance, which only compounds the difficulty of SLDS adequacy evaluation:

"From a commercial perspective, the success of a spoken dialogue system is only slightly related to technical matters. I make this somewhat bizarre pronouncement on the basis of first-hand practical experience. The key to commercial success is marketing: how a system is advertised to the end-users, how the system presents the company to those end-users, and how smoothly errors are handled. I have, for example, seen trial systems with a disgracefully low word accuracy score receiving a user satisfaction rating of
around 95%. I have also seen technically excellent systems being removed from service due to negative user attitudes.” [Norman Fraser, personal communication.]

Other open research issues include:
- how to evaluate portability of systems across application domains;
- comparative performance and adequacy evaluation across SLDSs for different tasks. [Hirschman and Thompson 1996]

This paper presents a partial scheme for the evaluation of dialogue components and integrated SLDSs. It is based on the development and testing of the Danish dialogue system and includes suggested improvements, in terms of concepts, methods and tools, to the evaluation procedures that were actually applied during development and test of the system. Section 2 addresses evaluation of requirement specifications for SLDSs. Section 3 describes evaluation of dialogue model design. Section 4 describes evaluation of the integrated system. Section 5 concludes the paper. Evaluation of speech recognition and understanding components, and of language and speech generation components will not be discussed in what follows.

The Danish dialogue system is a ticket reservation system for Danish domestic flights. The system runs on a PC with a DSP board and is accessed over the telephone. It is a walk-up-and-use application. It understands speaker-independent continuous spoken Danish with a vocabulary of about 500 words. The system is mixed-initiative, using system-directed domain communication and user-initiated, keyword-based meta-communication. The prototype runs in close-to-real-time. The system is a representative example of advanced state-of-the-art systems. Comparable SLDSs are found in [Aust and Oerder 1995, Cole et al. 1994, Eckert et al. 1995, Peckham 1993].

2. REQUIREMENT SPECIFICATION

The purpose of requirements specification is to list all the agreed requirements which the envisaged system should meet. There is no method which can ensure a complete and sufficient requirements specification. The craft and skills of experienced system developers are needed to make a qualified evaluation of imposed requirements. SLDS development and evaluation is still a relatively new field and there is no complete understanding of all the ingredients of SLDSs and their mechanisms of interaction. This adds to the difficulties of making a proper evaluation of an SLDS requirements specification. In the following we present experiences with establishing a requirements specification for the Danish dialogue system and proposals for its evaluation.

2.1 Realism criteria

The process of establishing a requirements specification for the Danish dialogue system was semi-realistic. The objective was to develop a realistic, application-oriented research prototype rather than a real application. This meant that we did not have real customers to talk to. However, we did have contact to a travel agency where we made interviews with travel agents and recordings of human-human reservation and information dialogues. The aim was to create a system which was realistic in the sense that it should meet, as far as possible, the needs and desiderata of potential customers. The system should offer economic advantage to potential customers and the choice of domain and technology should be reasonable in view of potential demands for SLDSs applications. For instance, it turned out to be a condition for launching the Danish dialogue project within the domain of telephone-based flight ticket reservation and information that a Danish parallel to the French Minitel did not exist at the time. Had such a system been in place, we had probably either chosen a different domain of application or a multi-modal approach which included speech input/output. Another result of our considerations of application realism was that the system should be able to run on a PC so that Danish travel agencies could easily afford the needed hardware. Had we chosen more powerful equipment, the performance constraints on the system would have been less severe.

2.2 Feasibility and usability

The feasibility and usability constraints on the system to be developed may be illustrated as follows. Since the system should be accessed over the telephone, real-time performance was considered mandatory for the system to be usable. In the context of the chosen hardware, and given the limited capabilities that could be expected from the speech recogniser, the real-time requirement gave rise to additional constraints on active vocabulary size and user utterance length. Furthermore, because of limited project resources the system vocabulary size was set to about 500 words although this was likely to be insufficient given the chosen domain of application. This constraint, of course, would be meaningless in a commercial development context. In addition to real-time performance, the main usability constraints were: sufficient task domain coverage, robustness, natural forms of language and dialogue, and dialogue flexibility.
2.3 Explicit requirements representation

As illustrated above, requirements behave as interacting constraints on the design process. This makes it desirable to create and maintain an explicit representation of the design space as it develops. If this is not being done, risks are that proper conclusions may fail to be drawn from interacting constraints with the result that the designers set out to what is in fact an internally conflicting task. We used the Design Space Development/Design Rationale (DSD/DR) approach to explicitly represent the evolving design space [Bernsen 1993b]. Several of the requirements mentioned above are represented in the DSD frame in Figure 1. A DSD frame represents the design space structure and designer commitments at a given point during system design. A series of DSD frames thus provides a series of snapshots of the developing design process. A DR frame represents the reasoning about a particular design problem (cf. Figure 5 in Section 4). It discusses the design options, constraint trade-offs and solutions considered and argues why a particular solution was chosen. Typically, there will be several DRs acting as links between two consecutive DSD frames. When combined with DR representations, DSD makes design space context and constraints explicit in support of reasoning, traceability and re-use.

We have had positive experience with using a DSD/DR representation in designing the Danish dialogue system. However, other methods of representation may be used instead. It is recommended to create an explicit requirements representation from the beginning of an SLDS development project. This is good engineering practice although often not followed with the result that is hard or even impossible to keep track of the design decisions that have been made and why they were made.

---

DSD No. N
A. General constraints and criteria

Overall design goal:
Spoken language dialogue system prototype operating via the telephone and capable of replacing a human operator;

General feasibility constraints:
Limited machine power available;

Scientific and technological feasibility constraints:
Limited capability of current speech and natural language processing;
Open research questions, e.g. research in dialogue theory;

Designer preferences:

Realism criteria:
The artifact should be preferable to current technological alternatives;
The system should run on machines which could be purchased by a travel agency;
The artifact should be tolerably inferior to the human it replaces, i.e., it should be acceptable by users while offering travel agencies financial advantage;

Functionality criteria:
 Usability criteria:
Maximize the naturalness of user-interaction with the system;
Constraints on system naturalness resulting from trade-offs with system feasibility have to be made in a principled fashion based on knowledge of users in order to be practicable by users;

B. Application of constraints and criteria to the artifact within the design space:

Collaborative aspects:
Organisational aspects:
System aspects:
500 words vocabulary;
Max 100 words in active vocabulary;
Limited speaker-independent recognition of continuous speech;
Close-to-real-time response;
Sufficient task domain coverage;

Interface aspects:
Spoken telephone dialogue;

Task aspects:
User tasks:
Obtain information on and perform booking of flights between two specific cities;
Use single sentences (or max. 10 words);
Use short sentences (average 3-4 words);

System tasks:
User aspects:
User experience aspects:

C. Hypothetical issues:

Is a vocabulary of 500 words sufficient to capture the sublanguage vocabulary needed in the task domain?

D. Documentation:

E. Conventions:
DSD No. (n) indicates the number of the current DSD specification.

---

Figure 1. DSD representation which shows some major requirements for the Danish dialogue system. The actual DSDs constructed during the Wizard of Oz phase can be seen in [Bernsen 1993b].
2.4 Evaluation of specific SLDS requirements

If speech input and/or output are being considered for the application to be developed, evaluation is needed of whether speech is suited for the application given the evolving requirements specification. In case of a multimodal system one should also consider how well speech combines with other modalities considered for the system.

Whether speech is well-suited or not depends on properties such as the task, its structure and complexity and on whether the requirements derived from these properties are compatible with other requirements on, e.g., budget, time, reliability and technology. As mentioned above, the flight reservation and information tasks were found well-suited for a speech application. However, we had insufficient knowledge at the time for estimating the structure and complexity of the tasks as well as the resulting demands on the user-system dialogue. We thus began by designing mixed-initiative dialogue for reservation of flight tickets, change of reservation and information on departures, fares and travel conditions, and performed a series of Wizard of Oz (WOZ) experiments (Section 3). However, it turned out during the WOZ experiments that mixed-initiative dialogue was not feasible given the hard constraint on active vocabulary size, cf. Figure 1. Furthermore, change of reservation and, in particular, the information task which consisted of many different sub-tasks that could be combined in arbitrary order, were not well-suited for system-directed dialogue. For these reasons, the information task was never implemented. The change of reservation task which might have been feasible, with some difficulty, in system-directed dialogue, was not implemented because of resource limitations. With more knowledge early in the design process about task types and the dialogue types required by different task types, the information task might have been excluded much earlier. This task would have been evaluated as being non-feasible due to the conflict between the minimum requirements expressed in Figure 2 and the requirements specification.

Figure 2 was developed on the basis of our dialogue model design. Note that the figure is incomplete in several respects: it excludes systems that do not have speech (input) understanding, such as voice response systems and ‘speech typewriters’; it does not consider the speaker-dependent/speaker-independent distinction; and user-directed dialogue needs more treatment. We have primarily compared relatively complex system-directed and mixed-initiative dialogue based on the distinction between well-structured and ill-structured tasks. Well-structured tasks have a stereotypical structure that prescribes which information needs to be exchanged between the dialogue partners to complete the task and, possibly, roughly in which order this is done. Such tasks may be acceptably managed through system-directed dialogue. Complex ill-structured tasks contain a large number of optional sub-tasks and hence are ill-suited for system-directed dialogue. Knowing, e.g., that a user wants travel information, does not help the system know what to offer and in which order. In such cases, some amount of user-directed dialogue or mixed-initiative dialogue would appear necessary to allow an acceptable minimum of usability.

2.5 Test criteria

Together with the requirements specification, performance and adequacy evaluation criteria should be established for the system to be developed. Early performance test criteria for the Danish dialogue system were the average and maximum user utterance lengths and the vocabulary size. We later discovered that we also needed a measurement for user initiative, cf. the discussion above. As a rough measure the number of user questions was used, cf. Section 3. Transaction success rate is a prime candidate adequacy evaluation criterion (Sections 3 and 4). Another possible criterion is the nature and number of interaction problems in a controlled scenario-based benchmark test (see Section 3). Subjective evaluation vehicles, such as questionnaires and interviews, are needed in addition to objective measures but it is very difficult to specify in advance the “scoring levels” that should be attained in questionnaires and interviews.

3. DIALOGUE DEVELOPMENT

Today’s dialogue model design for SLDSs development is largely based on empirical techniques, such as the WOZ experimental prototyping method in which a person simulates (part of) the system to be designed [Fraser and Gilbert 1991] and, for simple dialogues, implement-test-and-revise procedures based on emerging development platforms. These techniques mainly build on designers’ common sense, experience and intuition, and on trial and error. Whether WOZ is preferable to implement-test-and-revise depends i.a. on dialogue complexity and task domain and on risk and cost of implementation failure. WOZ is a costly method. However, by producing data material on the interaction between a (fully or partially) simulated system and its users it provides the basis for early tests of the system and hence also for testing the coverage and adequacy of requirements. A number of different tests may be carried out on the material produced by WOZ experi-
<table>
<thead>
<tr>
<th>Task type</th>
<th>Task type</th>
<th>Task type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small and simple tasks</td>
<td>Larger well-structured tasks</td>
<td>Larger ill-structured tasks</td>
</tr>
<tr>
<td>Limited domains</td>
<td>Limited domains</td>
<td>Limited domains</td>
</tr>
<tr>
<td><strong>Dialogue type</strong></td>
<td><strong>Dialogue type</strong></td>
<td><strong>Dialogue type</strong></td>
</tr>
<tr>
<td>Single-word dialogue</td>
<td>System-directed dialogue</td>
<td>Mixed-initiative dialogue</td>
</tr>
<tr>
<td><strong>Dialogue elements needed</strong></td>
<td><strong>Other technology needed</strong></td>
<td><strong>Other technology needed</strong></td>
</tr>
<tr>
<td>Either system or user initiative</td>
<td>Isolated word recognition</td>
<td>Continuous speech recognition</td>
</tr>
<tr>
<td>Limited system feedback</td>
<td>Small vocabulary</td>
<td>Medium-sized vocabulary</td>
</tr>
<tr>
<td></td>
<td>No syntactic and semantic analysis</td>
<td>Syntactic and semantic analysis</td>
</tr>
<tr>
<td></td>
<td>Look-up table of command words</td>
<td>Very limited handling of discourse phenomena</td>
</tr>
<tr>
<td></td>
<td>No handling of discourse phenomena</td>
<td>Representation of domain facts and rules, i.e. expert knowledge within the domain</td>
</tr>
<tr>
<td></td>
<td>Representation of domain facts, i.e. a database</td>
<td>Pre-recorded speech</td>
</tr>
<tr>
<td></td>
<td>Pre-recorded speech</td>
<td>Pre-recorded speech</td>
</tr>
</tbody>
</table>

**Figure 2.** Increased task complexity requires more sophisticated dialogue to maintain an acceptable level of habitability. This again requires more and better technologies and increases the demands on dialogue theory and on the elements supporting the dialogue model. The figure shows minimum requirements.

There is currently no agreement on which tests to carry out. We distinguish between three types of evaluation as mentioned in Section 1: diagnostic evaluation, performance evaluation and adequacy evaluation [Hirschman and Thompson 1996]. We shall also distinguish between objective evaluation and subjective evaluation. Diagnostic evaluation and performance evaluation are based on objective evaluation whereas adequacy evaluation include both objective and subjective evaluation.

The dialogue model for the Danish dialogue system was iteratively developed by the WOZ method. Seven WOZ iterations involving a total of 24 users were performed to produce the dialogue model which was implemented [Dybkjær et al. 1993]. The WOZ experiments produced a transcribed corpus of 125 scenario-based, task-oriented human-machine dialogues corresponding to approximately seven hours of spoken dialogue. We also collected a corpus of 25 human-human reservation dialogues in a travel agency. However, we only used these dialogues to obtain information on the order in which the needed reservation details were achieved by the travel agent. At this level human-human dialogue parallels may serve as input to systems design. But the dialogues as such are much different from corresponding human-machine dialogues. Human-machine dialogues have to be much simpler than human-human dialogues because otherwise the system cannot handle them. Moreover, it is well-known that people tend to address computers in a way which is different from how they address humans, perhaps because of the systems’ limited capabilities. For these reasons only human-machine data, such as those obtained through WOZ, are really reliable as a basis for a dialogue model.
3.1 Diagnostic evaluation

A major concern during WOZ is to detect and diagnose problems of user-system interaction. Eventually, we used two approaches, both based on the dialogue model representation, to systematically discover such problems. The dialogue model used in the WOZ experiments was represented as a complex state transition network that had system output in the nodes and expected contents of user utterances along the edges, cf. Figure 3.

The matching approach

One approach was to match, prior to each WOZ iteration, the scenarios to be used against the current dialogue model representation in order to discover and remove potential dialogue design problems. If a deviation from the state transition network occurred during the matching process, this would indicate a potential dialogue design problem which should be removed, if possible. Significantly, many problems were discovered analytically through these scenario-based designer walkthroughs of the dialogue model. This seems to be typical of dialogue model development and illustrates the need for a tool, such as a set of design guidelines, which could help designers prevent such problems from occurring.

The plotting approach

The second approach was to plot the transcribed dialogues onto the current dialogue model representation in order to systematically detect dialogue design problems from the interaction problems that occurred. As in the first approach, state transition network deviations indicated potential dialogue design problems. Deviations were marked and their causes analysed whenever the dialogue model was revised, if necessary. Figure 3 shows an annotated sub-graph from WOZ6. The annotation shows that the user expected the system to confirm the commitments made. When it became clear that the system was not going to provide confirmation, the subject asked for it. The following dialogue fragment provides the background for the subject's deviation from the dialogue model. The subject has made a change to a flight reservation. After the user has stated the change, the dialogue continues (S is the simulated system, U is the user):

S7: Do you want to make other changes to this reservation?
U7: No, I don't.
S8: Do you want anything else?
U8: Ah no, ... I mean is it okay then?
S9: [Produces an improvised confirmation of the change made.]
U9: Yes, that's fine.
S10: Do you want anything else?

From this point the dialogue finishes as expected. Analysis convinced us that the dialogue model had to be revised in order to prevent the occurrence of the user-initiated clarification meta-communication observed in U8, which the implemented system would be incapable of understanding. In fact, the WOZ6 dialogue model can be seen to have violated the following dialogue design principle: Be fully explicit in communicating to users the commitments they have made. As a result, system confirmation of changes of reservation was added to the WOZ7 sub-graph on change of reservation.

Figure 3. A plotted END sub-graph from WOZ6. The boldfaced loop that deviates from the graph path shows unexpected user dialogue behaviour which may reveal a dialogue design problem. The circled number (3) refers to the point in the CHANGE sub-graph from which the experimenter jumped to the END sub-graph. The deviation is annotated with numbered reference (in italics) to the relevant transcribed utterances and a description of the deviation. S refers to the system and U to the user.

Design guidelines

Many design errors were detected through use of the two above approaches. However, it would have been preferable if we could have prevented these errors from occurring in the first place. Towards the end of WOZ we started to develop a tool which could serve the purpose of preventing interaction problems and which could be used no matter if WOZ is used or not.

All problems of interaction uncovered during WOZ were analysed and represented as violations of principles of cooperative spoken human-machine dialogue. Each problem was considered a case in which the system, in addressing the user, had violated a principle of cooperative dialogue. The principles were made explicit, based on the problems analysis. The WOZ corpus analysis led to the identification of
14 principles of cooperative spoken human-machine dialogue based on analysis of 120 examples of user-system interaction problems [Bernsen 1993a]. Each of the 14 principles was accompanied by a justification which served the additional purpose of clarifying its meaning and scope. If the principles were observed in the design of the system’s dialogue behaviour, we assumed, this would serve to reduce the occurrence of user dialogue behaviour that the system had not been designed to handle.

The 14 principles of cooperative spoken human-machine dialogue were refined and achieved their present formulation as shown in Figure 4 through comparison with Grice’s Cooperative Principle and maxims for cooperative human-human dialogue [Bernsen et al. 1996a]. Only SP10 and SP11 (on meta-communication) and the last part of GP10 were added later as a result of using the principles in analysing the dialogue corpus from the user test of the implemented system, cf. Section 4. The distinction between principle and aspect (Figure 4) is useful because an aspect represents the property of dialogue addressed by a particular principle. A generic principle may subsume one or more specific principles which specialise the generic principle to certain classes of phenomena. Although subsumed by generic principles, we believe that specific principles are useful to SLDS dialogue design. The principles are used by manually evaluating if each system utterance in isolation as well as in context violates any of the generic or specific principles. If it does, it is a potential source for communication failure which should be removed.

So far we have not had the opportunity to use the principles as design guidelines in an SLDS development process. However, we have successfully used them for evaluation purposes during the user test, as will be discussed in more detail in Section 4.

3.2 Performance evaluation

Between each of the seven WOZ experiments the dialogue model was evaluated and, based on the results, modified in order to achieve improved performance. The performance tests measured average and maximum utterance lengths, vocabulary size and convergence, and user initiative was roughly measured in terms of number of user questions. We also compared the results to those of earlier WOZ iterations in order to measure progress. The utterance lengths were eventually reduced to meet the requirements. The vocabulary, however, although sufficiently small within each iteration did not show convergence. Convergence towards zero of the cumulative word type/token ratio would indicate that the vocabulary size is sufficiently large for the application and that new users cannot be expected to introduce new words. However, as expected, a 300 words vocabulary turned out to be insufficient.

The early WOZ iterations allowed free mixed-initiative dialogue. We gradually transferred dialogue initiative to the system by letting the system ask questions of the user, thereby reducing the average user utterance length and the active vocabulary size. Much effort went into achieving a dialogue structure which corresponded to the one that users would expect based on their experiences from human-human reservation dialogues. Again this served to prevent the occurrence of user initiative. The domain dialogue was eventually made completely system-directed which turned out to be necessary in order to meet the constraint on active vocabulary size (Figure 1). Had we had the knowledge expressed in Figure 2 at the start of the WOZ experiments, we would have known already then that mixed-initiative domain communication would not be feasible.

3.3 Adequacy evaluation

We did not perform any objective adequacy evaluation of the WOZ material. However, it may be recommended to at least carry out evaluation of the transaction success. Although only based on simulated human-machine dialogue, such an evaluation may still provide valuable information on dialogue acceptability. The system should be implemented only when minimum requirements on transaction success have been met. Transaction success could thus serve as a stop criterion for WOZ. Transaction success is discussed in more detail in Section 4.

Subjective evaluation parameters

As user satisfaction is not just achieved through technically excellent systems and cannot be sufficiently measured through objective evaluation, it is important to collect users’ opinions on the system being developed at the earliest possible. WOZ provides a good basis for collecting users’ opinions prior to system implementation, for instance through questionnaires and interviews. Questionnaires and interviews can be useful in identifying weaknesses that have been overlooked or cannot easily be identified through objective measurement. The difficulty with questionnaires and interviews is which questions to ask and how, and how to interpret the answers. Questionnaires also tend to be rigid, in particular if multiple choice is being used. If, on the other hand, questions are too open the risk is that people do not tell us what we would like to know. Also, people often do not like
<table>
<thead>
<tr>
<th>Group 1: Informativeness</th>
<th>GP1</th>
<th>Make your contribution as informative as is required (for the current purposes of the exchange).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SP1</td>
<td>Be fully explicit in communicating to users the commitments they have made.</td>
</tr>
<tr>
<td></td>
<td>SP2</td>
<td>Provide feedback on each piece of information provided by the user.</td>
</tr>
<tr>
<td></td>
<td>GP2</td>
<td>Do not make your contribution more informative than is required.</td>
</tr>
<tr>
<td>Group 2: Truth and evidence</td>
<td>GP3</td>
<td>Do not say what you believe to be false.</td>
</tr>
<tr>
<td></td>
<td>GP4</td>
<td>Do not say that for which you lack adequate evidence.</td>
</tr>
<tr>
<td>Group 3: Relevance</td>
<td>GP5</td>
<td>Be relevant, i.e. Be appropriate to the immediate needs at each stage of the transaction.</td>
</tr>
<tr>
<td>Group 4: Manner</td>
<td>GP6</td>
<td>Avoid obscurity of expression.</td>
</tr>
<tr>
<td></td>
<td>GP7</td>
<td>Avoid ambiguity.</td>
</tr>
<tr>
<td></td>
<td>SP3</td>
<td>Provide same formulation of the same question (or address) to users everywhere in the system’s dialogue turns.</td>
</tr>
<tr>
<td></td>
<td>GP8</td>
<td>Be brief (avoid unnecessary prolixity).</td>
</tr>
<tr>
<td></td>
<td>GP9</td>
<td>Be orderly.</td>
</tr>
<tr>
<td>Group 5: Partner asymmetry</td>
<td>GP10</td>
<td>Inform the dialogue partners of important non-normal characteristics which they should take into account in order to behave cooperatively in dialogue. Ensure the feasibility of what is required of them.</td>
</tr>
<tr>
<td></td>
<td>SP4</td>
<td>Provide clear and comprehensible communication of what the system can and cannot do.</td>
</tr>
<tr>
<td></td>
<td>SP5</td>
<td>Provide clear and sufficient instructions to users on how to interact with the system.</td>
</tr>
<tr>
<td>Group 6: Background knowledge</td>
<td>GP11</td>
<td>Take partners’ relevant background knowledge into account.</td>
</tr>
<tr>
<td></td>
<td>GP12</td>
<td>Take into account possible (and possibly erroneous) user inferences by analogy from related task domains.</td>
</tr>
<tr>
<td></td>
<td>SP6</td>
<td>Separate whenever possible between the needs of novice and expert users (user-adaptive dialogue).</td>
</tr>
<tr>
<td></td>
<td>SP7</td>
<td>Take into account legitimate partner expectations as to your own background knowledge.</td>
</tr>
<tr>
<td></td>
<td>GP13</td>
<td>Provide sufficient task domain knowledge and inference.</td>
</tr>
<tr>
<td>Group 7: Repair and clarification</td>
<td>GP13</td>
<td>Initiate repair or clarification meta-communication in case of communication failure.</td>
</tr>
<tr>
<td></td>
<td>SP9</td>
<td>Provide ability to initiate repair if system understanding has failed.</td>
</tr>
<tr>
<td></td>
<td>SP10</td>
<td>Initiate clarification meta-communication in case of inconsistent user input.</td>
</tr>
<tr>
<td></td>
<td>SP11</td>
<td>Initiate clarification meta-communication in case of ambiguous user input.</td>
</tr>
</tbody>
</table>

Figure 4. The generic and specific principles of cooperativity in dialogue. Each specific principle is subsumed by a generic principle. The left-hand column characterises the aspect of dialogue addressed by each principle.

to spend time on writing about what they liked and did not like about the system. This is much easier to communicate in an interview. In interviews, however, subjects are rarely asked precisely the same questions in precisely the same way. This makes it even more difficult to compare user answers. In addition, people tend to express what they like and what they dislike in rather different ways.

In the last two WOZ iterations, we asked subjects to fill in a questionnaire after their interaction with the simulated system. In this questionnaire, users were first asked about their background, including how familiar they were with the task, with voice-response systems and with systems understanding speech. They were then asked a number of multiple choice questions on the dialogue system. For each question they were asked to tick off one in five boxes on a scale from negative to positive, for instance ‘difficult’ versus ‘easy’. The questions were the following: how was it to solve the tasks; what do you think of the number of errors made by the
system; how was it to make corrections; how do you find the system now; would you prefer to call a travel agent or the system if you had the choice; what do you think of dialogue systems like this in the future; how well-prepared were you to use the system; how do you find the present system: rigid or flexible, stimulating or boring, frustrating or satisfactory, efficient or inefficient, desirable or undesirable, reliable or unreliable, complicated or simple, impolite or friendly, predictable or unpredictable, acceptable or not acceptable (all with the possibility of five choices). Finally, users were asked to provide free-style comments on whether something ought to be changed in the way in which users should address the system, what they liked about the system and what they did not like. On the average, users found the system rigid and boring and would prefer to talk to a human travel agent. Otherwise they were positive. The negative evaluation on the three points mentioned was not surprising given the rigid system-directed dialogue. The really valuable knowledge from a systems design point of view, however, was rather obtained through the free-style answers. In these, users would sometimes be very specific about what annoyed them when they used the system, thus providing us with clues to improvements.

We also interviewed users on the phone immediately after their interaction with the system. However, this was only to ask if they believed the system was real and to debrief them on the experiment.

4. THE IMPLEMENTED SYSTEM

The implemented system was subjected to the same tests as was the simulated system. In addition we measured transaction success and, based on the developed design principles presented in Section 3, we made a detailed analysis and evaluation of dialogue design problems. Also a blackbox test was carried out whereas a glassbox test was left out to save resources.

4.1 Glassbox and blackbox

There is no general agreement on the definitions of glassbox and blackbox tests. By a glassbox test we shall understand a test in which the internal system representation can be inspected. The test should make use of test suites that will activate all loops and conditions of the program being tested. The relevant test suites are constructed by the system programmer(s) along with an indication of which program parts the test suites are supposed to activate. Via test print-outs in all loops and conditions it is possible to check which ones were actually activated.

In a blackbox test only input to and output from the program are available to the evaluator. How the program works internally is made invisible. Test suites are constructed on the basis of the requirements specification and along with an indication of expected output. Expected and actual output are compared when the test is performed and deviations must be explained. Either there is a bug in the program or the expected output was incorrect. Bugs must be corrected and the test run again. The test suites should include fully acceptable as well as borderline cases to test if the program reacts reasonably and does not break down in case of errors in input. Ideally, and in contrast to the glassbox test suites, the blackbox test suites should not be constructed by the system programmer who implemented the system since s/he may have difficulties in viewing the program as a black box.

The dialogue model resulting from the seven WOZ iterations was implemented, as was the rest of the system. The dialogue model was, as mentioned, not subjected to a glassbox test whereas a blackbox test was carried out. The implemented dialogue model was embedded in the entire system except for the recogniser which was disabled to allow reconstruction of errors. Internal communication between system modules was logged in logfiles. We created a number of test suites all containing user input for one or more reservations of one-way tickets and return tickets with or without discount.

A test suite always had to include an entire reservation involving several interdependent system and user turns. In a query-answering system a task will often only involve one user turn and one system turn. Hence one may ask a question and simply from the system answer determine if the system functions correctly for the test case. In a task such as ticket reservation which involves several turns, the system’s reactions to the entire sequence of turns must be correct. An apparently correct system reaction, as judged from the system’s immediate reaction, may turn out to have been partly wrong when we inspect the sequence of interdependent system reactions. Hence to test our dialogue model it was not sufficient to test, e.g., isolated transactions concerning customer numbers, possible destinations or a selection of dates. Also the combinations of the test suites had to be considered. Furthermore, because each test reservation can only test a limited amount of cases we had to create a long series of test reservations.

The blackbox test was not entirely exhaustive. However, the test did reveal a number of problems. Some of these were due to disagreements between the dialogue model specification and the implementation. But the majority of problems were such that
had not been taken into account during specification. Each of the discovered problems were represented in a DR-frame along with a discussion of possible solutions, cf. Figure 5.

Resources were not available for implementing solutions to all the problems discovered. It was therefore considered, for each problem, how time consuming the implementation of a solution would be and how important it was. The solutions which were implemented influenced not only the implementation but also the specification including the order of the dialogue structure. This again implied that the test suites had to be revised to bring them in agreement with the specification. The revised dialogue model was blackbox tested with the revised test suites. Bugs were corrected but no major new unknown problems were revealed.

4.2 User test with a simulated recogniser

A controlled user test of the implemented system was carried out with a simulated speech recogniser [Berne et al. 1995]. A wizard key in the users’ answers into the simulated recogniser. The simulation ensured that typos were automatically corrected and that input to the parser corresponded to an input string which could have been recognised by the real

<table>
<thead>
<tr>
<th>Design Project: P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepares DSD No. 8</td>
</tr>
<tr>
<td>DR No. 6</td>
</tr>
<tr>
<td>Date: 24.5.94</td>
</tr>
</tbody>
</table>

**Design problem:** No price information

**Commitments involved**

1. It should be possible for users to fully exploit the system’s task domain knowledge when they need it.

2. Avoid superfluous or redundant interactions with users (relative to their contextual needs).

**Justification**

Only some users are interested in getting information on the price. Professional users loose time on an extra dialogue turn if they are asked whether they want it. On the other hand, for users wanting the price information this may be very important.

**Options**

1. Provide full price breakdown information at the end of a reservation task.

2. Ask users if they want to know the price of their reserved tickets.

3. Always inform users about the total price of their reservation (but not its breakdown into the prices of individual tickets).

**Resolution: Option 3**

There is a clash between the two design commitments because of the existence of different needs in the user population. Option 3 was identified and selected as a compromise between the two relevant design commitments. Option 3 does not require extra turn taking but mentions the price briefly.

**Comments**

Since P1 already computes the price it will be easy also to output this information to the user. It would be a possibility to allow the user to obtain additional price information (a breakdown into the prices of individual tickets) via the help function (see DR 12).

**Time estimate for developing and implementing solution**

Less than 1 day.

**Links to other DRs**

12 (help).

**Documentation**

**Insert into next DSD frame**

Option 3.

**Status**

Do the implementation.

Figure 5. A DR-frame for one of the problems detected during the blackbox test of the implemented dialogue model.
recogniser. The recognition accuracy would be 100% as long as users expressed themselves in accordance with the vocabulary and grammars known to the system. Otherwise, the simulated recogniser would turn the user input into a string which only contained words and grammatical constructions from the recogniser’s vocabulary and rules of grammar.

A user test is meant to test if the system functionality expected by the user is present. A user test may be carried out as a controlled test or as a field test. In a controlled user test the users need not be those who will actually use the final system. However, it is recommended to select the test subjects from the target group to ensure that they have a relevant background. The background may influence the way in which people interact with the system. The tasks to be carried out (scenarios) are not selected by the participants in the controlled user test. To ensure a reasonable coverage of the test and representativeness of scenarios and to bring it as close to benchmarking as possible, the scenario selection should ideally be made by an independent panel according to certain guidelines on, i.e., who should select the scenarios, their coverage of system functionality, number of scenarios per user and number of users. The panel should include end-users as well as system developers. A field distribution problem attaches to all results of controlled user tests. The frequency of different tasks across the domain of application may be different in real life from that imposed in the controlled user test. This may affect the frequency of different interactions problems.

In a field test real end-users are used as testers. The system to be tested is inserted in the environment in which it is supposed to work and is used. This means that the tasks carried out will be real-life tasks but will not necessarily be a representative selection unless the duration of the field test is very long. For reasense test the option of a field test will not always be available due to the missing customer. However, a controlled test may be preferable anyway because it allows an evaluation close to benchmarking.

The controlled user test of the Danish dialogue system was based on 20 different scenarios which had been designed by the system designers to enable exploration of all aspects of the task structure. Since the flight ticket reservation task is a well-structured task, it was possible to extract from the task structure a set of sub-task components, such as number of travellers, age of traveller, and discount vs. normal fare, any combination of which should be handled by the dialogue system. The scenarios were generated from systematically combining these components.

Twelve external subjects who had never tried the system and who represented the target group, mostly professional secretaries, participated in the user test. Subjects conducted the dialogues over the telephone in their normal work environments in order to make the situation as realistic as possible. Each subject received an introductory letter, a brochure briefly describing the system, four scenarios and a questionnaire. Totally subjects were given 50 particular tasks based on 48 scenarios. A task consists in ordering one or more tickets for one route. The number of recorded dialogues was 57 because subjects sometimes reiterated a failed dialogue and eventually succeeded with the task. A dialogue is one path through the dialogue structure. As in the WOZ experiments, all dialogues were recorded and transcribed. In addition, all transactions between the individual system modules were logged. The test material was subjected to diagnostic evaluation, performance evaluation, and adequacy evaluation, the latter involving objective as well as subjective evaluation.

4.3 Diagnostic evaluation

The analysis of the user test corpus aimed at detecting problems of dialogue interaction. We did not use a state transition network representation of the implemented dialogue model as we had done for the simulated one. Instead, we used the following approach which not only considers deviations from the dialogue structure but also details of formulation. Based on the dialogue structure, a template was built which contained the system’s questions. For each scenario, normative system questions and user answers were filled into the template. The key contents of the actual dialogues were then plotted into the template [Dybkjær et al. 1996]. An example is shown in Figure 6. A translation from the Danish of the problematic part of the dialogue is provided in the following:

S: Monday January 16. At which time?
U: 7:20.
S: There is no departure at 7:20. The closest other red departure is at 10:50. Do you want this departure?
U: No thanks. [The above four turns are repeated another three times with the only difference that the user adds "change" after "no thanks" in two cases.]
S: At which time?
U: Change.
S: The journey starts on Monday January 16. Is that correct?
U: Yes.
S: At which time
U: 7:20.
S: There is no departure at 7:20. The closest other
Comparison between normative and actual system and user utterances and analysis of deviations which sometimes required use of the logged transactions between the system modules, led to the identification of three main classes of interaction problems: (1) linguistic problems, (2) problems of dialogue interaction and (3) other problems, such as cases of system breakdown. Dialogue interaction problems split into (a) dialogue design problems and (b) user errors [Bernsen et al. 1996b]. Only dialogue design problems are discussed in the following.

Each identified dialogue design problem was categorised according to which cooperativity principle, cf. Figure 4, had been violated. The following GPs and SPs were found violated at least once: GPs 1, 3, 5, 6, 7, 10, 11, 12, 13 and SPs 2, 4, 5, 6, 8, 10, 11. For classification purposes the principles were modified to express the cooperativity problem they represented, e.g. GP1: “Make your contribution as informative as is required (for the current purposes of the exchange)” was turned into “The system provides less information than required”. Each problem was described in terms of its symptom (S), a diagnosis (D) was made and a cure (C) proposed, cf. Figure 7.

<table>
<thead>
<tr>
<th>Scenario: G-1-4-a User: 2 Date: January 13 1995</th>
<th>System questions</th>
<th>Normative user answers</th>
<th>Actual user answers</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>System already known</td>
<td>no / yes / -</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Customer number</td>
<td>3</td>
<td>3</td>
<td>GP1, SP10</td>
<td></td>
</tr>
<tr>
<td>Number of travellers</td>
<td>1</td>
<td>1</td>
<td>GP1</td>
<td></td>
</tr>
<tr>
<td>ID-numbers</td>
<td>2</td>
<td>2</td>
<td>GP1</td>
<td></td>
</tr>
<tr>
<td>Departure airport</td>
<td>Aalborg</td>
<td>Aalborg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arrival airport</td>
<td>Copenhagen</td>
<td>Copenhagen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Return journey</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interested in discount</td>
<td>no / yes</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day of departure (out)</td>
<td>January 16</td>
<td>Monday (January 16)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hour of departure (out)</td>
<td>7:20</td>
<td>7:20 (no departure)</td>
<td>SP5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7:20 (no departure)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no, change [does not want one from list; change not caught by system]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7:20 (no departure)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no [does not want one from list]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7:20 (no departure)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no [does not want one from list]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>change [hour of departure]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>yes [out-day is January 16]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7:20 (no departure)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>change [hour of departure]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>yes [hour of departure is 7:20]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day of departure (home)</td>
<td>January 16</td>
<td>Same day (January 16)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hour of departure (home)</td>
<td>17:45</td>
<td>17:45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivery</td>
<td>airport / send</td>
<td>airport</td>
<td></td>
<td></td>
</tr>
<tr>
<td>More</td>
<td>no</td>
<td>no</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6. Key contents of the expected (normative) and actual user-system exchanges in the dialogue G14a. In the third column key contents of the system’s replies are indicated in parentheses unless they can be derived from the explanatory comments in square brackets. GP means generic principle and SP means specific principle.
S: Are you particularly interested in discount?
U: Yes please....
S: There is no departure at 7:20. The closest other red departure is at 10:50.
D: The system provides insufficient information. It does not tell that there is a blue departure at 7:20.
C: The system should provide sufficient information, e.g. by telling that there is no red departure but that there is a blue departure at the chosen hour.

Figure 7. Violation of GP1 in dialogue G14a. The system response is incomplete. It withholds important information and is therefore misleading. S is system and U is user.

The user test also served as a test of our cooperative principles and confirmed their broad coverage with respect to cooperative spoken user-system dialogue. Almost all of the 119 individual dialogue design problems identified in the user test material could be ascribed to violations of the cooperative principles. Only three additions had to be made to the principles established during WOZ. Two specific principles of meta-communication were added, i.e. SP10 and SP11 in Figure 4. Since meta-communication had not been simulated during WOZ and the WOZ corpus therefore contained few examples of meta-communication, this came as no surprise.

More interestingly, we had to add a modification to GP10, namely that it should be feasible for users to do what they are asked to do. For instance, in its introduction the system asks users to use the keywords 'change' and 'repeat' for meta-communication purposes and to answer the system's questions briefly and one at a time. Despite the introduction, a significant number of violations of those instructions occurred in the user test. For instance, users attempted to make changes through full-sentence expressions rather than by saying 'change'. Almost all of these cases led to misunderstanding or non-understanding. These violations of clear system instructions were initially categorised as user errors. However, upon closer analysis they were re-categorised as dialogue design problems. Although the system has clearly stated that it has non-normal characteristics due to which users should modify their natural dialogue behaviour, this is not cognitively possible for many users.

4.4 Performance evaluation
For the performance evaluation we measured the same parameters as in the WOZ experiments, i.e. the average and maximum utterance lengths, vocabulary size, and user initiative. The average user utterance length was still well within the required limits. However, the prescribed maximum user utterance length was exceeded in 17 cases. 10 of these utterances were produced by the same subject. Particularly in the first dialogue, this subject tended to repeat an utterance if the system did not answer immediately. The majority of long utterances, both for this subject and in general, was caused by user-initiated corrections which did not make use of the keyword 'correct' but were expressed in free style by users. Two long utterances were produced by subjects who took over the initiative when asked 'Do you want anything else?'. This question was clearly too open.

As predicted, the system's vocabulary was insufficient. The test corpus showed 51 out-of-vocabulary word types.

Subjects sometimes took over the initiative by providing more information than had been asked for and in four cases they asked questions. One question was asked because the subject had misread the scenario text. The three remaining user questions all concerned available departure times. This is not surprising since departure times constitute a type of information which users often do not have in advance but expect to be able to obtain from the system. When users lack information, the reservation task tends to become informed reservation and hence an ill-structured task.

4.5 Adequacy evaluation
Adequacy evaluation should include measurement of transaction success. There is still no standard definition of "transaction success" [Giachin 1996]. In the Danish dialogue system we defined successes as reservations carried out according to the scenario specification or according to the user's mistaken interpretation of the scenario. As failures were counted reservations in which the user failed to get what was asked for even if this was due to an error committed by the user. Based on this definition, the task transaction success for the user test was 86% in that seven tasks were counted as transaction failures. One of the failures was exclusively caused by a user who did not listen to the system's feedback and a second transaction failure was caused by a combination of a system problem (SP11) and a user error. The five remaining transaction failures were caused by system problems, i.e. violations of the principles GP5, SP2, SP4, SP5 and SP11, cf. Figure 4.

Misinterpretation of scenarios such as not asking for discount or ordering a one-way ticket instead of a return ticket is not a problem in real life. Nevertheless the situation is not desirable in a controlled user test since users carry out another scenario than they were asked to do which may affect system evaluation. A scenario which is not carried
out may result in that part of the dialogue model remains untested.

An open question is whether transaction failures exclusively caused by user errors should be counted as failures or not. One may ask to which extent it is reasonable to blame the system for a failure.

One could also consider to use the result of the diagnostic evaluation of number and types of interaction problems as part of the adequacy evaluation. However, the problem is how to specify quantitative criteria in advance. It is not obvious how many and which types of interaction problems could be accepted.

Transaction success and number of interaction problems are not sufficient for measuring adequacy. For example, one cannot draw conclusions on user satisfaction from the transaction success rate nor from the number of interaction problems encountered.

**Subjective evaluation parameters**

To learn more on user satisfaction a subjective evaluation is needed. Therefore, also in the user test subjects were asked to fill in a questionnaire and received a telephone interview after interaction with the system. The questionnaire was very similar to one given to Woz subjects. Only three questions had been added: how was the system's speech; what do you think of the language used; was the system fast or slow. Output quality was rated high whereas subjects did not find that they could use free natural language. They found the system slow. These results are not surprising in view of the requirement to use keywords in initiating meta-communication, the missing sub-vocabulary parts, and the fact that the test used a bionic wizard system.

Many of the multiple choice answers were very similar to those from the Woz questionnaires.

Positive improvements over Woz 7 could be seen on acceptability, efficiency, usefulness and ease of task performance. There were also improvements in the evaluation of stimulatingness and preference of the system over a human travel agent but both were still low. The main reasons probably were the rigid dialogue structure and, in particular for the latter, the (correct) impression that the system has limited capabilities and cannot cope with non-routine matters.

There were drops in the positive evaluation on two important parameters, namely flexibility and ease of making corrections. The low evaluation on flexibility is probably due to the rigid, system-directed dialogue structure and the restriction to keywords for meta-communication. The negative development with respect to ease of making corrections is probably due to the fact that misunderstandings were not simulated in Woz 7. This meant that hardly any user-initiated meta-communication was required. In addition, the use of keywords for making corrections does not form part of the natural human linguistic skills.

Again as in Woz, some useful and specific comments were given in reply to the open questions in the questionnaire. Although many subjects tended to write only one or two brief comments, a few subjects had bothered to write detailed and very useful replies.

In the telephone interview immediately after their interaction with the system users were asked the following four questions: How was it to talk to the system; what is your immediate impression of the system (specific problems/advantages); do you think the system was real; would you be interested in trying the system with the real recogniser. Like the free-style comments in the questionnaire, the telephone interviews provided important information on users' opinions of the system. The opinions expressed in the interviews were in accordance with the multiple choice answers in the questionnaire but contributed explanations of why the users held their opinions.

We did not ask the users to assign priority to their critical comments on the system. However, even if we had done this and modified the system accordingly, there would be no guarantee that users would then be satisfied with the system. User satisfaction is a conglomerate of many parameters, objective as well as subjective ones, cf. Section 1, and users may not even be aware of all the parameters which are important to them.

5. **CONCLUDING DISCUSSION**

This paper has addressed issues of SLDS evaluation as regards requirement specification, dialogue model design and the implemented and integrated system. Methods and tools used or developed during evaluation of the dialogue model of the Danish dialogue system were presented and discussed. The presentation was structured in terms of distinction between diagnostic evaluation, performance evaluation and adequacy evaluation. In particular adequacy evaluation is difficult because it is not exclusively based on objective evaluation. Some of the test subjects were not at all interested in speaking to a computer system. This attitude may or may not change as speech systems become more common. Most people would probably be willing to use a speech understanding system provided that it is sufficiently attractive. However, what is considered attractive may vary from person to person. To some users, for instance, a mediocre system may become highly at-
tractive if they receive a price reduction on tickets booked via this system.

Research is obviously needed on methods and tools which can support the three types of evaluation discussed. More research is also needed on aspects of evaluation which have not been addressed above. These include comparative systems evaluation, SLDS customisability evaluation, SLDS maintainability evaluation, strengths and limitations of speech functionality for different tasks, users, environments etc., speech and multimodality, and ergonomic aspects of speech applications.

ACKNOWLEDGEMENTS

The Danish dialogue system was developed in collaboration between the Center for Person-Kommunikation at Aalborg University (speech recognition, grammar), the Centre for Language Technology, Copenhagen (grammar, parsing), and the Centre for Cognitive Science, Roskilde University (dialogue and application design and implementation, human-machine aspects). The project was supported by the Danish Research Councils for the Technical and the Natural Sciences. We gratefully acknowledge the support.

REFERENCES


