Spoken Dialogue Technology: Enabling the Conversational User Interface

MICHAEL F. McTEAR

University of Ulster

Spoken dialogue systems allow users to interact with computer-based applications such as databases and expert systems by using natural spoken language. The origins of spoken dialogue systems can be traced back to Artificial Intelligence research in the 1950s concerned with developing conversational interfaces. However, it is only within the last decade or so, with major advances in speech technology, that large-scale working systems have been developed and, in some cases, introduced into commercial environments. As a result many major telecommunications and software companies have become aware of the potential for spoken dialogue technology to provide solutions in newly developing areas such as computer-telephony integration. Voice portals, which provide a speech-based interface between a telephone user and Web-based services, are the most recent application of spoken dialogue technology. This article describes the main components of the technology—speech recognition, language understanding, dialogue management, communication with an external source such as a database, language generation, speech synthesis—and shows how these component technologies can be integrated into a spoken dialogue system. The article describes in detail the methods that have been adopted in some well-known dialogue systems, explores different system architectures, considers issues of specification, design, and evaluation, reviews some currently available dialogue development toolkits, and outlines prospects for future development.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Natural language, Voice I/O*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Discourse, Speech recognition and synthesis*

General Terms: Human Factors

Additional Key Words and Phrases: Dialogue management, human computer interaction, language generation, language understanding, speech recognition, speech synthesis

1. INTRODUCING SPOKEN DIALOGUE TECHNOLOGY

The "conversational computer" has been the goal of researchers in speech technology and artificial intelligence (AI) for more than 30 years. A number of large-scale research programs have addressed this goal, including the DARPA Communicator Project, Japan's Fifth Generation program, and the European Union's ESPRIT and Language Engineering programs. The

---

Author's address: School of Information and Software Engineering, University of Ulster, Shore Road, Newtownabbey BT37 0QB, Northern Ireland, UK; email: MF.McTear@ulst.ac.uk.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
©2002 ACM 0360-0300/02/0300-0001 $5.00

impression of effortless spontaneous conversation with a computer has been fostered by examples from science fiction such as HAL in *2001: A Space Odyssey* or the computer on the Star Ship Enterprise. It is only recently, however, that spoken language interaction with computers has become a practical possibility both in scientific as well as in commercial terms. This is due to advances in speech technology, language processing, and dialogue modeling, as well as the emergence of faster and more powerful computers to support these technologies. Applications such as voice dictation and the control of appliances using voice commands are appearing on the market and an ever-increasing number of software and telecommunications companies are seeking to incorporate speech technology into their products. It is important, however, to be aware of the limitations of these applications. Commonly statements are made in sales and marketing literature such as *Talk to your computer as you would talk to your next-door neighbor* or *Teach your computer the art of conversation*. However, the technologies involved would not be sufficient to enable a computer to engage in a natural conversation with a user. Voice dictation systems provide a transcription of what the user dictates to the system, but the system does not attempt to interpret the user's input or to discuss it with the user. Command-and-control applications enable users to perform commands with voice input that would otherwise be performed using the keyboard or mouse. The computer recognizes the voice command and carries out the action, or reports that the command was not recognized. No other form of dialogue is involved. Similar restrictions apply to most other forms of voice-based system in current use.

Spoken dialogue systems, on the other hand, can be viewed as an advanced application of spoken language technology. Spoken dialogue systems provide an interface between the user and a computer-based application that permits spoken interaction with the application in a relatively natural manner. In so doing, spoken dialogue systems subsume most of the major fields of spoken language technology, including speech recognition and speech synthesis, language processing, and dialogue management.

The aim of the current survey is to describe the essential characteristics of spoken dialogue technology at a level of technical detail that should be accessible to computer scientists who are not specialists in speech recognition and computational linguistics. The survey provides an overview for those wishing to research into or develop spoken dialogue systems, and hopefully also for those who are already experienced in this field. Most published work to date on spoken dialogue systems tends to report on the design, implementation, and evaluation of individual systems or projects, as would be expected with an emerging technology. The present paper will not attempt to survey the growing number of spoken dialogue systems currently in existence but rather will focus on the underlying technologies, using examples of particular systems to illustrate commonly occurring issues.[1]

1.1. Overview of the Paper

The remainder of the paper is structured as follows. In the next section spoken dialogue systems are defined as computer systems that use spoken language to interact with users to accomplish a task. Dialogue systems are classified in terms of different control strategies and some examples are presented in Section 3 that illustrate this classification and give a feel for the achievements as well as the limitations of current technology. Section 4 describes the components of a spoken dialogue system—speech recognition, language understanding, dialogue management, external communication, response generation, and text-to-speech synthesis.

---

[1] Inevitably there are omissions, in some cases of well-known and important systems, but this is unavoidable, as the aim is not to provide a comprehensive review of dialogue systems but to focus on the general issues of the technology. Interested readers can follow up particular systems in the references provided at the end of the survey and in Appendix A.

The key to a successful dialogue system is the integration of these components into a working system. Section 5 reviews a number of architectures and dialogue control strategies that provide this integration. Methodologies to support the specification, design, and evaluation of a spoken dialogue system are reviewed in Section 6. Particular methods have evolved for specifying system requirements, such as user studies, the use of speech corpora, and Wizard-of-Oz studies. Methods have also been developed for the evaluation of dialogue systems that go beyond the methods used for evaluation of the individual elements such as the speech recognition and spoken language understanding components. This section also examines some current work on guidelines and standards for spoken language systems. A recent development is the emergence of toolkits and platforms to support the construction of spoken dialogue systems, similar to the toolkits and development platforms that are used in expert systems development. Some currently available toolkits are reviewed and evaluated in Section 7. Finally, Section 8 examines directions for future research in spoken dialogue technology.

2. SPOKEN DIALOGUE SYSTEMS—A DEFINITION

Spoken dialogue systems have been defined as computer systems with which humans interact on a turn-by-turn basis and in which spoken natural language plays an important part in the communication [Fraser 1997]. The main purpose of a spoken dialogue system is to provide an interface between a user and a computer-based application such as a database or expert system. There is a wide variety of systems that are covered by this definition, ranging from question-answer systems that answer one question at a time to "conversational" systems that engage in an extended conversation with the user. Furthermore, the mode of communication can range from minimal natural language, consisting perhaps of only a small set of words such as the digits 0–9 and the words *yes* and *no*, through to large vocabulary systems supporting relatively free-form input. The input itself may be spoken or typed and may be combined with other input modes such as DTMF (touch-tone) input, while the output may be spoken or displayed as text on a screen, and may be accompanied by visual output in the form of tables or images.

Spoken dialogue systems enable casual and naive users to interact with complex computer applications in a natural way using speech. Current IVR (Interactive Voice Response) systems limit users in what they can say and how they can say it. However, users of speech-based computer systems often do not know exactly what information they require and how to obtain it—they require the support of the computer to determine their precise requirements. For this reason it is essential that speech-based computer systems should be able to engage in a dialogue with users rather than simply respond to predetermined spoken commands. At the same time spoken dialogue systems are more restricted than conversational computers in that their conversational topics are limited, usually to a single domain such as flight enquiries.

Spoken dialogue systems can be classified into three main types, according to the methods used to control the dialogue with the user:

(1) finite state- (or graph-) based systems;
(2) frame-based systems; and
(3) agent-based systems.

The type of dialogue control strategy used has a bearing on how the system accomplishes two of its main tasks: processing the user's input and recovering from errors.

2.1. Finite State-Based Systems

In a finite state-based system the user is taken through a dialogue consisting of a sequence of predetermined steps or states. Most commercially available spoken dialogue systems use this form of dialogue control strategy. The dialogue flow is specified as a set of dialogue states with transitions denoting various alternative paths through the dialogue graph. The system maintains control of the dialogue, produces prompts at each dialogue state, recognizes (or rejects) specific words and phrases in response to the prompt, and produces actions based on the recognized response. The following is an example of an interaction with a basic finite state-based system in which the system verifies the user's input at each state of the dialogue:

System: *What is your destination?*
User: *London.*
System: *Was that London?*
User: *Yes.*
System: *What day do you want to travel?*
User: *Friday.*
System: *Was that Sunday?*
User: *No.*
System: *What day do you want to travel?*

Generally systems with state-based dialogue control restrict the user's input to single words or phrases that provide responses to carefully designed system prompts. A major advantage of this form of dialogue control is that the required vocabulary and grammar for each state can be specified in advance, resulting in more constrained speech recognition and language understanding. However, the disadvantage is that these systems restrict the user's input to predetermined words and phrases, making correction of misrecognized items difficult as well as inhibiting the user's opportunity to take the initiative and ask questions or introduce new topics. If augmented with a natural language component, a state-based system can accept more natural input in the form of sentences or partial sentences. Furthermore, with a simple data structure such as a form or frame that keeps track of which information the user has provided and what the system still needs to know, a more flexible and more natural dialogue flow is possible. Verification can also be delayed until the system has gathered all the required information. The Nuance demo banking system to be described in Section 3 is an example of a state-based system with these additional functionalities.

2.2. Frame-Based Systems

In a frame- (or template-) based system the user is asked questions that enable the system to fill slots in a template in order to perform a task such as providing train timetable information. In this type of system the dialogue flow is not predetermined but depends on the content of the user's input and the information that the system has to elicit. For example:

System: *What is your destination?*
User: *London.*
System: *What day do you want to travel?*
User: *Friday.*
System: *What is your destination?*
User: *London on Friday around 10 in the morning.*
System: *I have the following connection....*

In the first example the user provides one item of information at a time and the system performs rather like a state-based system. However, if the user provides more than the requested information, as in the second example, the system can accept this information and check if any additional items of information are required before searching the database for a connection. Frame-based systems function like production systems, taking a particular action based on the current state of affairs. The questions and other prompts that the system should ask can be listed, along with the conditions that have to be true for a particular question or prompt to be relevant. Some form of natural language input is required by frame-based systems to permit the user to respond more flexibly to the system's prompts, as in the second example. Natural language is also required to correct errors of recognition or understanding by the system. Generally, however, it is sufficient for the system to be able to recognize the main concepts in the user's utterance. The Philips train timetable information, to be described in Section 3, is an example of a frame-based system.

2.3. Agent-Based Systems

Agent-based or AI systems are designed to permit complex communication between the system, the user, and the underlying application in order to solve some problem or task. There are many variants on agent-based systems, depending on what particular aspects of intelligent behavior are included in the system. The following dialogue, taken from Sadek and de Mori [1998], illustrates a dialogue agent that engages in mixed-initiative cooperative dialogue with the user:

User: *I'm looking for a job in the Calais area. Are there any servers?*

System: *No, there aren't any employment servers for Calais. However, there is an employment server for Pas-de-Calais and an employment server for Lille. Are you interested in one of these?*

In this example the system's answer to the user's request is negative. But rather than simply responding *No*, the system attempts to provide a more cooperative response that might address the user's needs.

In agent-based systems communication is viewed as interaction between two agents, each of which is capable of reasoning about its own actions and beliefs, and sometimes also about the actions and beliefs of the other agent. The dialogue model takes the preceding context into account, with the result that the dialogue evolves dynamically as a sequence of related steps that build on each other. Generally there are mechanisms for error detection and correction, and the system may use expectations to predict and interpret the user's next utterances. These systems tend to be mixed initiative, which means that the user can take control of the dialogue, introduce new topics, or make contributions that are not constrained by the previous system prompts. For this reason the form of the user's input cannot be determined in advance as consisting of a set number of words, phrases, or concepts, and, in the most complex systems, a sophisticated natural language understanding component is required to process the user's utterances. The Circuit-Fix-It-Shop system, to be presented in Section 3, is an example of one type of agent-based system. Other types will be discussed in Section 5.

2.4. Verification

In addition to the different levels of language understanding required by different types of dialogue system, there are also different methods for verifying the user's input. In the most basic state-based systems, in which user input is restricted to single words or phrases elicited at each state of the dialogue, the simplest verification strategy involves the system confirming that the user's words have been correctly recognized. The main choice is between confirmations associated with each state of the dialogue (i.e., every time a value is elicited the system verifies the value before moving on to the next state), or confirmations at a later point in the transaction. The latter option, which is illustrated in the example from the Nuance banking system in Section 3, provides for a more natural dialogue flow. The more natural input permitted in frame-based systems also makes possible a more flexible confirmation strategy in which the system can verify a value that has just been elicited and, within the same utterance, ask the next question. This strategy of implicit verification is illustrated in the example from the Philips train timetable information system in Section 3. Implicit verification provides for a more natural dialogue flow as well as a potentially shorter dialogue, and is made possible because the system is able to process the more complex user input that may arise when the user takes the initiative to correct the system's misrecognitions and misunderstandings. Finally, in agent-based systems, more complex methods of verification (or "grounding") are required along with decisions as to how and when the grounding is to be achieved. Verification will be discussed in greater detail in Section 4.3.2, and some examples of verification strategies can be seen in the examples presented in Section 3.

2.5. Knowledge Sources for Dialogue Management

The dialogue manager may draw on a number of knowledge sources, which are sometimes referred to collectively as the dialogue model. A dialogue model might include the following types of knowledge relevant to dialogue management:

A dialogue history: A record of the dialogue so far in terms of the propositions that have been discussed and the entities that have been mentioned. This representation provides a basis for conceptual coherence and for the resolution of anaphora and ellipsis.

A task record: A representation of the information to be gathered in the dialogue. This record, often referred to as a form, template, or status graph, is used to determine what information has not yet been acquired (see Section 5.2). This record can also be used as a task memory [Aretoulaki and Ludwig 1999] for cases where a user wishes to change the values of some parameters, such as an earlier departure time, but does not need to repeat the whole dialogue to provide the other values that remain unchanged.

A world knowledge model: This model contains general background information that supports any commonsense reasoning required by the system, for example, that Christmas day is December 25.

A domain model: A model with specific information about the domain in question, for example, flight information.

A generic model of conversational competence: This includes knowledge of the principles of conversational turn-taking and discourse obligations—for example, that an appropriate response to a request for information is to supply the information or provide a reason for not supplying it.

A user model: This model may contain relatively stable information about the user that may be relevant to the dialogue—such as the user's age, gender, and preferences—as well as information that changes over the course of the dialogue, such as the user's goals, beliefs, and intentions.

These knowledge sources are used in different ways and to different degrees according to the dialogue strategy chosen. In the case of a state-based system these models, if they exist at all, are represented implicitly in the system. For example, the items of information and the sequence in which they are acquired are predetermined and thus represented implicitly in the dialogue states. Similarly, if there is a user model, it is likely to be simple and to consist of a small number of elements that determine the dialogue flow. For example, the system could have a mechanism for looking up user information to determine whether the user has previous experience of this system. This information could then be used to allow different paths through the system (for example, with less verbose instructions), or to address user preferences without having to ask for them.

Frame-based systems require an explicit task model as this information is used to determine what questions still need to be asked. This is the mechanism used by these systems to control the dialogue flow. Generally the user model, if one exists, would not need to be any more sophisticated than that described for state-based systems. Agent-based systems, on the other hand, require complex dialogue and user models as well as mechanisms for using these models as a basis for decisions on how to control the dialogue. Information about the dialogue history and the user can be used to constrain how the system interprets the user's subsequent utterances and to determine what the system should say and how it should

Table I. Dialogue Control Strategies

Features/Dialogue control strategy	State-based	Frame-based	Agent-based
Input	Single words or phrases	Natural language with concept spotting	Unrestricted natural language
Verification	Explicit confirmation—either of each input or at end of transaction	Explicit and implicit confirmation	Grounding
Dialogue model	Information state represented implicitly in dialogue states Dialogue control represented explicitly with state diagram	Explicit representation of information states Dialogue control represented with control algorithm	Model of system's intentions, goals, and beliefs. Dialogue history, context
User model	Simple model of user characteristics or preferences	Simple model of user characteristics or preferences	Model of user's intentions, goals, and beliefs

be said. These sorts of modeling involve representations of discourse structure, of intentions, goals, and beliefs, and of dialogue as a collaborative activity. Various approaches to dialogue and user modeling in agent-based systems will be exemplified and discussed in later sections of this survey.

2.6. Summary

This section has provided a classification of spoken dialogue systems in terms of the type of dialogue control strategy used and has shown how the choice of dialogue control strategy is closely related to the form of the user's input as well as the use of particular error recovery mechanisms. These points are summarized in Table I. The next section presents three examples of spoken dialogue systems that illustrate these different control strategies.

3. EXAMPLES OF SPOKEN DIALOGUE SYSTEMS

3.1. State-Based Dialogue Control: The Nuance Automatic Banking System

The Nuance automatic banking system enables callers to conduct transactions over the telephone such as paying a bill, transferring funds between accounts, or obtaining an account balance. The system conducts a dialogue in English with the user, who can speak to the system using relatively unrestricted speech. It should be stressed that the following dialogues are based on recorded interactions with a demo system available in the UK (available at: +44 1223 516959), used here to illustrate some important aspects of state-based dialogue control. The shortcomings that are presented in some of the examples would have been addressed in a fully developed commercial application.

System: *What would you like to do?*
User: *Pay a bill.*
System: *What company would you like to pay?*
User: *Midland Bank.*
System: *How much would you like to pay?*
User: *One hundred and twenty-five pounds fifty-seven pence.*
System: *What date would you like the payment to be made on?*
User: *Monday.*
System: *You want to pay HSBC one hundred fifty-five pounds and fifty-seven pence on April 5th? Is this correct?*
User: *Yes.*
System: *Your payment will be sent. Would you like to pay another bill?*

In order to complete the transaction the system has to elicit three specific values from the caller—payee, amount, transfer date. These are confirmed and the transaction is completed with a return to the sub-menu for bill payments. Some points

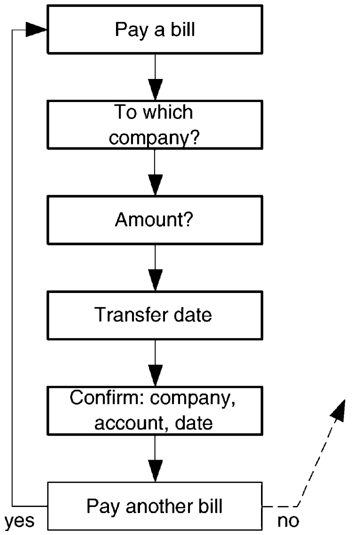


Fig. 1. Dialogue flow for payment of a bill.

worth noting in this example are that the system is able to process phrases specifying currency amounts, is able to transform relative dates such as *Monday* into absolute dates such as *April 5th*, and is able to handle multiple synonyms (*Midland Bank* or *HSBC*).

The dialogue flow, which is system-directed with a predetermined sequence of questions, can be modeled as a set of interconnected dialogue states, as shown in Figure 1.

Transferring amounts between accounts involves a different type of transaction in which the system can accept more complex natural language input from the caller, thus reducing the number of questions to be asked and the time required for the transaction:

System: *What would you like to do?*
User: *I would like to transfer two hundred and fifty pounds from my current to my savings account.*
System: *You want to transfer two hundred fifty pounds from current to savings. Is this correct?*
User: *Yes.*

This transaction is also modeled using dialogue states combined with multiple slot filling. Additional transitions need to be provided depending on which combinations of the values of amount, source account, and destination account are provided by the user in the natural language input. Problems can arise if this functionality is not available throughout the dialogue, as illustrated in the following excerpt:

System1: *What company would you like to pay?*
User1: *Abbey National.*
System2: *How much would you like to pay?*
User2: *One hundred pounds next Monday.*
System3: *What date would you like the payment to be made on?*
User3: *Next Monday.*
System4: *You want to pay Abbey National one hundred pounds and sixty-one pence on April 5th? Is this correct?*
User4: *No.*
System5: *Would you like to change the company, amount, or date?*
User5: *Amount.*
System6: *How much would you like to pay?*
User6: *One hundred pounds.*
System7: *You want to pay Abbey National one hundred pounds on April 5th? Is this correct?*
User7: *Yes.*

In User2 the user provides an overinformative answer, specifying not only the amount requested in System2 but also the date of the transaction. The system's next question (System3) indicates that either the date was misrecognized or that the system is not set up for multiple slot filling at this dialogue state. Whatever the reason, the system goes on to ask the question required at the next dialogue state, that is, for the date of the transaction. Thus the user is requested to supply information that had just been given in the previous response. Moreover, as shown in System4, the system has recognized *next Monday* not as a date but as part of the amount, that is, as *and sixty-one pence*. As shown in System5–User7, the system is able to recover from this error by getting the user to reconfirm the amount.

A final example illustrates some limitations on the system's mechanisms for error recovery:

System: *Would you like to change the company, amount, or date?*
User: *The company and the date.*
System: *Please say company, amount, or date.*

From this example it can be seen that the system is unable to correct more than one error at a time. Thus, while the strategy of leaving the confirmation of the three items of information until they have all been elicited will normally result in a more natural and faster transaction, it may lead to a frustrating and rather unnatural dialogue where more than one item has to be changed.

The Nuance banking system is an advanced spoken dialogue system that employs a state-based dialogue control in conjunction with natural language understanding and multiple slot filling. Given good recognition rates the system is likely to perform well and complete transactions efficiently and effectively. These sample interactions were chosen carefully to illustrate some potential problems that are likely to arise when a state-based system is extended to provide more sophisticated functionality. The examples would appear to indicate that the multiple slot filling afforded by the system's natural language processing is not available at each dialogue state. Although this can be easily remedied, a further complication is introduced as now the system has to maintain a dialogue model of which slots have been filled so as to direct the dialogue down alternative paths. With small numbers of slots this is not a problem, but with an increasing number of slots the combinations of state transitions soon reach unmanageable proportions. This is particularly the case when further transitions are added to deal with verification of the slot values. Thus, while state-based dialogue control is useful for implementing simple transactions, introducing more sophisticated functionality may require additional mechanisms such as an explicit dialogue model that obscure the semantics of the state-based control. This issue will be discussed in greater detail in Section 5. The next example will show how a frame-based system is able to handle these issues more effectively.

3.2. Frame-Based Dialogue Control: The Philips Automatic Train Timetable Information System

The Philips Automatic Train Timetable Information System (available at: +49 241 604020) provides information over the telephone about train connections between 1,200 German cities. The system conducts a natural dialogue in German with the user, who can speak to the system using fluent, unrestricted speech. The aim of the dialogue is to enable the system to construct an appropriate database query that retrieves the information required by the user and then to present that information in a suitable form. This system falls within a class of similar systems concerned with the provision of information, such as travel schedules, banking, stocks and shares, and weather forecasts. The German version of the system is a research prototype, which is not connected with the official information service provided by German Rail (Deutsche Bahn AG). The following is an example of a dialogue with the Philips Automatic Train Timetable Information System, translated from the original German [Aust et al. 1995]:

System1: *Good morning. This is the automatic timetable information service. How can I help you?*
User1: *I need a connection from Hamburg to Munich.*
System2: *When would you like to go from Hamburg to Munich?*
User2: *Next Friday at eight o'clock.*
System3: *So you want to travel on Sunday the 20th of November at eight o'clock a.m.?*

User3: *No, on Friday.*
System4: *So you want to travel on Friday the 18th of November at eight o'clock a.m.?*
User4: *Yes.*
System5: *There is the following connection: With IntercityExpress 793, departure from Hamburg at 8:02, arrival in Munich at 14:05. Do you want me to repeat the connection?*
User5: *No, but I need a train for the return trip on Sunday.*
System6: *At what time would you like to go from Munich to Hamburg on Sunday the 20th of November?*
User6: *I have to arrive in Hamburg late in the evening.*
System7: *So you want to arrive in Hamburg between nine p.m. and eleven p.m.?*
User7: *Yes.*
System8: *There are three connections. This is the first one: With IntercityExpress 880, departure from Munich at 15:54, arrival in Hamburg at 21:58. Do you want the next connection?*
User8: *No, thanks, that one is fine.*
System9: *Do you need yet another connection?*
User9: *No, thank you.*
System10: *We wish you a pleasant journey. Thank you for calling. Good bye.*

It is informative to examine the dialogue flow in this example. The purpose of the system is to guide the user toward providing all the required values—departure city, destination city, departure time, and departure date, as well as ascertaining other values, such as whether the time is a.m. or p.m. and whether the time refers to the departure or arrival. These values are acquired over the course of the dialogue and once they have been acquired the system can execute a suitably constrained database query and provide one or more connections.

This system could have been implemented using state-based control in which the system would progress through a number of states eliciting and verifying the required values. Alternatively the system could have been implemented as a *question-answer* or *natural language database* system in which the user would have been required to provide all the required parameters in one utterance, which could then have been translated directly into a database query. However, one of the aims of the designers of this system was to enable a more flexible interaction in which the user would not be constrained either to input one value at a time or to input all the values within one utterance. This flexibility is necessary as it cannot be determined in advance what a user might know regarding the information required to make a valid query. For example, the system may need to know if the user wishes to travel on an Intercity train, requires a train with restaurant facilities, and so on. If the user is not aware of all the possibilities, the system has to issue relevant queries and elicit suitable values in order to find the best connection.

A second aspect of dialogue flow concerns the sequencing of the system's questions. There should be a logical order to the questions. This order may be largely determined by what information is to be elicited in a well-structured task such as a travel information enquiry. The disadvantage of a state-based approach combined with natural language processing capabilities is that users may produce overinformative answers that provide more information than the system has requested at that point. In the Philips example at System2–User2, the system's more open-ended prompt *When would you like to go from Hamburg to Munich?* is ambiguous in that it can allow the user to supply departure time or date or both—as happens in User2. Even with a more constrained prompt such as *On which day would you like to go from Hamburg to Munich?* the user might supply both date and time. A system that followed a predetermined sequence of questions might then ask *At what time would you like to go* *from Hamburg to Munich?*—an unacceptable question as the time has already been given. The Philips system uses a status graph to keep track of which slots have already been filled. This mechanism will be described in greater detail in Section 5.

A close examination of the dialogue also shows that the system is able to deal with recognition errors and misunderstandings. For example, in System3 the system attempts to confirm the departure date and time but has misrecognized the departure date and is corrected by the user in User3. More subtly, the system uses different strategies for confirmation. In System2 an implicit confirmation request is used in which the values for departure city and destination provided by the user in User1 are echoed back within the system's next question, which also includes a request for the value for the departure date and/or time. If the system's interpretation is correct, the dialogue can proceed smoothly to the next value to be obtained and the user does not have to confirm the previous values. Otherwise, if the system has misunderstood the input, the user can correct the values before answering the next question. Conversely, an explicit confirmation request halts the dialogue flow and requires an explicit confirmation from the user. An example occurs in System3–User3 in which the system makes an explicit request for the confirmation of the departure date and time and the user corrects the date. The next exchange System4–User4 is a further example of an explicit confirmation request to verify the departure date and time.

One further aspect of the Philips system is its robustness. An example can be seen at System6–User6. In response to the system prompt for the departure time, the user does not provide a direct response containing the required time but states a constraint on the arrival time, expressed vaguely as *late in the evening*. The system is able to interpret this expression in terms of a range (*between 9 p.m. and 11 p.m.*) and to find an appropriate departure time that meets this constraint. More generally, the system is robust enough to be able to handle a range of different expressions for dates and times (e.g., *three days before Christmas, within this month*) and to be able to deal with cases of missing and contradictory information.

The provision of information such as train times is a typical application of spoken dialogue technology. Philips has developed a system with similar functionality for Swiss Rail, which has been an official part of Swiss Rail's information service since 1996. Public reaction to the system has been favorable, with over 80% of the people who used the service rating it as "excellent." Strik et al. [1996] reported on a project involving adaptation of the German system to the Dutch public transport network, while the European R&D project ARISE, which includes the Dutch, French, and Italian railway operators, builds on earlier European projects and on the Philips systems to provide more elaborate services with a multilingual component. The strengths and weaknesses of frame-based dialogue control are discussed further in Section 5.

3.3. Agent-Based Dialogue Control for Problem Solving—The Circuit-Fix-It Shop System

Problem-solving tasks usually involve some degree of co-operation between agents, particularly when one of the agents is unable to solve the problem alone. Cooperation may take the form of assigning particular subtasks to particular agents with the accomplishment of the subtasks contributing to the completion of the overall task. Such co-operative problem solving will require communication between the agents to report on the state-of-play of the problem solving and to supply the information required by the other agent. The next example illustrates collaborative problem solving in the Circuit-Fix-It Shop system [Smith and Hipp 1994]. Other agent-based systems will be discussed in Section 5.

The Circuit-Fix-It Shop system helps users to fix an electronic circuit by engaging in a spoken dialogue with the user. Communication between system and user

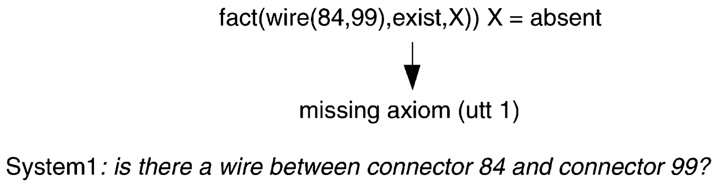

fact(wire(84,99),exist,X)) X = absent
missing axiom (utt 1)
System1: is there a wire between connector 84 and connector 99?

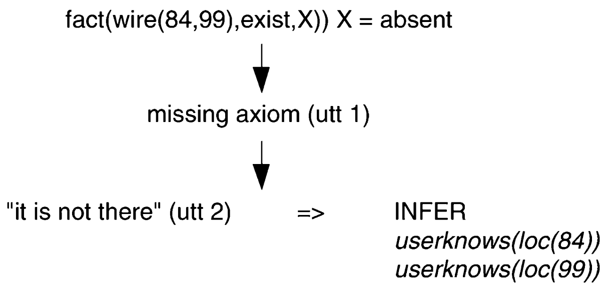
Fig. 2. Goal tree before utterance System1.

is motivated by the fact that neither participant is fully equipped to solve the problem alone. The system has all the necessary knowledge about the task but does not have the required sensory abilities to monitor the state of the world or to perform actions in the world. These are the concern of the human user who has also some knowledge of the task, which can range from that of a novice to that of an expert, depending on the individual user. The system employs user modeling—estimating the user's knowledge of particular objects and actions based on inferences from earlier exchanges in the dialogue—to determine the content of the information to be communicated to the user. Context-dependent expectations are also used to correct speech recognition hypotheses and to track when the user shifts to a different subdialogue. The following is an example of a dialogue between a user and the system:

System1: *Is there a wire between connector 84 and connector 99?*
User1: *It is not there.*
System2: *Add a wire between connector 84 and connector 99.*
User2: *I need help.*
System3: *Obtain a wire.*
User3: *Done.*
System4: *Connect metal of one free end of the wire to connector 84.*
User4: *How?*
System5: *Bend back connector 84.*
User5: *Done.*
System6: *Insert free end of wire into connector 84.*
User6: *Wire between 84 and 99 is connecting.*
System7: *What is the LED displaying?*

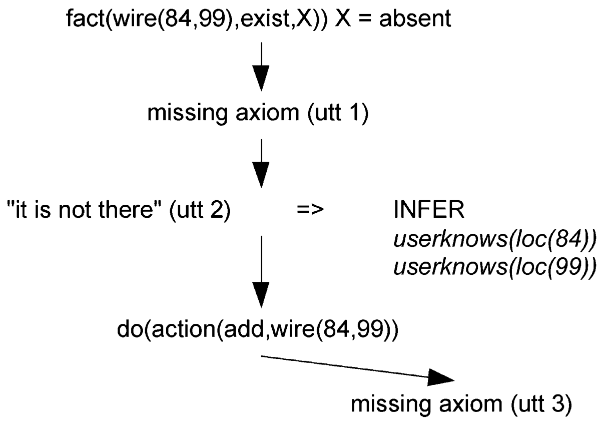
The dialogue evolves in the form of a proof, which can be illustrated using a goal tree. The goal tree represents the solution of the problem as it evolves dynamically. The system invokes rules to prove the goal in a top-down fashion—as in Prolog-style theorem proving. The proof may succeed using internally available knowledge, in which case no dialogue is required. However, the system is designed to deal with cases where the proof fails because the information required to complete the proof is not available to the system. In this case the system engages in dialogue with the user to obtain the missing information (described as "missing axioms") so that the proof can succeed.

At the beginning of the dialogue, the system does not know whether there is a wire between connector 84 and connector 99. As this is a missing axiom in the current proof, the system produces utterance System1 to ask the user. The state of the proof at this point is shown in the following goal tree displayed in Figure 2. The user confirms that the wire is missing. From this the system can infer that the user knows the location of the connectors and these facts are added to the user model. Figure 3 shows the current state of the goal tree. So that the current goal can be completed, the system instructs the user to add a wire between the connectors. This yields the goal tree shown in Figure 4. As the user does not know how to do this, a subgoal is inserted instructing the user on how to accomplish this task. This subgoal consists of the actions: locate connector 84, locate connector 99, obtain a wire, connect one end of wire to 84, and connect other end of wire to 99. These items are added to the goal tree depicted in Figure 5. However, as the user model contains the information that the user can



User1*: it is not there*

Fig. 3. Goal tree after utterance User1.



System2: *add a wire between connector 84 and connector 99*

Fig. 4. Goal tree after utterance System2.

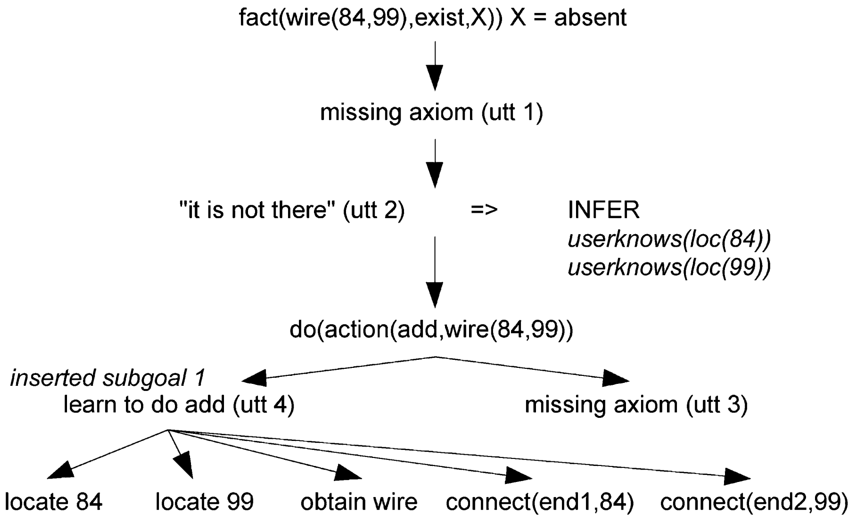


Fig. 5. Goal tree after utterance User2.

locate these connectors, instructions for the first two actions are not required and so the system proceeds with instructions for the third action, which is confirmed in User3, and for the fourth action. Here the user requires further instructions, which are given in System5 with the action confirmed in User5. At this point the user asserts that the wire between 84 and 99 is connecting, so that the fifth instruction to connect the second end to 99 is not required. A further missing axiom is discovered, which leads the system to ask what the LED is displaying (System7).

3.4. Summary

The examples presented in this section have illustrated three different types of dialogue control strategy. The selection of a dialogue control strategy determines the degree of flexibility possible in the dialogue and places requirements on the technologies employed for processing the user's input and for correcting errors. There are many variations on the dialogue strategies illustrated here, and these will be discussed in greater detail in Section 5. The next section will examine the component technologies of spoken dialogue systems.

4. COMPONENTS OF A SPOKEN DIALOGUE SYSTEM

A spoken dialogue system involves the integration of a number of components that typically provide the following functionalities [Wyard et al. 1996]:

Speech recognition: The conversion of an input speech utterance, consisting of a sequence of acoustic-phonetic parameters, into a string of words.

Language understanding: The analysis of this string of words with the aim of producing a meaning representation for the recognized utterance that can be used by the dialogue management component.

Dialogue Management: The control of the interaction between the system and the user, including the coordination of the other components of the system.

Communication with external system: For example, with a database system, expert system, or other computer application.

Response generation: The specification of the message to be output by the system.

Speech output: The use of text-to-speech synthesis or prerecorded speech to output the system's message.

These components are examined in the following subsections in relation to their role in a spoken dialogue system (for a recent text on speech and language processing, see Jurafsky and Martin [2000]).

4.1. Speech Recognition

The task of the speech recognition component of a spoken dialogue system is to convert the user's input utterance, which consists of a continuous-time signal, into a sequence of discrete units such as phonemes (units of sound) or words. One major obstacle is the high degree of variability in the speech signal. This variability arises from the following factors:

Linguistic variability: Effects on the speech signal caused by various linguistic phenomena. One example is coarticulation, that is, the fact that the same phoneme can have different acoustic realizations in different contexts, determined by the phonemes preceding and following the sound in question.

Speaker variability: Differences between speakers, attributable to physical factors such as the shape of the vocal tract as well as factors such as age, gender, and regional origin; and differences within speakers, due to the fact that even the same words spoken on a different occasion by the same speaker tend to differ in terms of their acoustic properties. Physical factors such as tiredness, congested airways due to a cold, and changes of mood have a bearing on how words are pronounced, but the location of a word within a sentence and the degree of emphasis it is given are also factors which result in intraspeaker variability.

Channel variability: The effects of background noise, which can be either constant or transient, and of the transmission channel, such as the telephone network or a microphone.

The speech recognition component of a typical spoken dialogue application has to be able to cope with the following additional factors:

Speaker independence: As the application will normally be used for a wide variety of casual users, the recognizer cannot be trained on an individual speaker (or small number of speakers) who will use the system, as is the case for dictation systems; instead, for speaker-independent recognition samples have to be collected from a variety of speakers whose speech patterns should be representative of the potential users of the system. Speaker-independent recognition is more error-prone than speaker-dependent recognition.

Vocabulary size: The size of the vocabulary varies with the application and with the particular design of the dialogue system. Thus a carefully controlled dialogue may constrain the user to a vocabulary limited to a few words expressing the options that are available in the system, while in a more flexible system the vocabulary may amount to more than a thousand words.

Continuous speech: Users of spoken dialogue systems expect to be able to speak normally to the system and not, for example, in the isolated word mode employed in some dictation systems. However, it is difficult to determine word boundaries in continuous speech since there is no physical separation in the continuous-time speech signal.

Spontaneous conversational speech: Since the speech that is input to a spoken dialogue system is normally spontaneous and unplanned, it is typically characterized by disfluencies, such as hesitations and fillers (for example, *amm* and *er*, false starts, in which the speaker begins one structure then breaks off midway and starts again, and extralinguistic phenomena such as coughing. The speech recognizer has to be able to extract from the speech signal a sequence of words from which the speaker's intended meaning can be computed.

The basic process of speech recognition involves finding a sequence of words, using a set of models acquired in a prior training phase, and matching these with the incoming speech signal that constitutes the user's utterance. The models may be word models, in the case of systems with a small vocabulary, but more typically the models are of units of sound such as phonemes or *triphones*, which model a sound as well as its context in terms of the preceding and succeeding sounds. The most successful approaches view this pattern-matching as a probabilistic process which has to be able to account both for *temporal variability*—due to different durations of the sounds resulting from differences in speaking rate and the inherently inexact nature of human speech—and *acoustic variability*—due to the linguistic, speaker, and channel factors described earlier. The following formula expresses this process:

$$\mathcal{W}^* = \operatorname*{argmax}_{w} P(O \mid W)P(W).$$

In this formula $\mathcal{W}^*$ represents the word sequence with the maximum *a posteriori* probability, while $\mathcal{O}$ represents the observation that is derived from the speech signal. Two probabilities are involved: $P(O \mid W)$, known as the *acoustic model*, which has been derived through a training process and which is the probability that a sequence of words $\mathcal{W}$ will produce an observation $\mathcal{O}$; and a *language model*, derived from an analysis of a language corpus giving the prior probability distribution assigned to the sequence of words $\mathcal{W}$.

The observation $\mathcal{O}$ comprises a series of vectors representing acoustic features of the speech signal. These feature vectors are derived from the physical signal, which is sampled and then digitally encoded. Perceptually important speaker-independent features are extracted and redundant features are discarded.

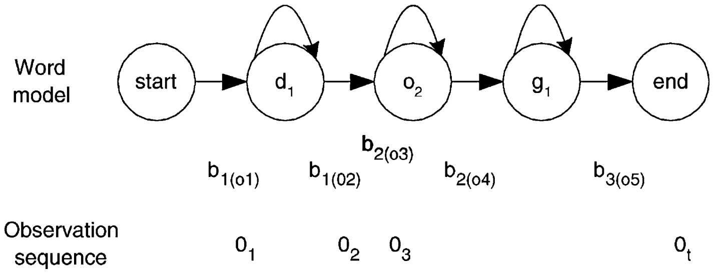


Fig. 6. A simple hidden Markov model.

Acoustic modeling is a process of mapping from the continuous speech signal to the discrete sounds of the words to be recognized. The acoustic model of a word is represented in hidden Markov models (HMMs), as in Figure 6. Each state in the HMM might represent a unit of sound, for example, the three sounds in the word *dog*. Transitions between each state, $A = a_{12}a_{13}\dots a_{n1}\dots a_{nn}$, represent the probability of transitioning from one state to the next and model the temporal progression of the speech sounds. Due to variability in the duration of the sounds, a sound may spread across several frames so that the model can take a loop transition and remain in the same state. For example, if there were five frames for the word *dog*, the states sequence S_1, S_1, S_2, S_2, S_3 might be produced, reflecting the longer duration of the sounds representing *d* and *o*. A hidden Markov model is doubly stochastic, as in addition to the transition probabilities the output of each state, $B = b_{i(ot)}$, is probabilistic. Instead of each state having a single unit of sound as output, all units of sound are potentially associated with each state, each with its own probability. The model is "hidden" because, given a particular sequence of output symbols, it is not possible to determine which sequence of states produced these output symbols. It is, however, possible to determine the sequence of states that has the highest probability of having generated a particular output sequence. In theory this would require a procedure that would examine all possible state sequences and compute their probabilities. In practice, because of the Markov assumption that being in a given state depends only on the previous state, an efficient dynamic programming procedure such as the Viterbi algorithm or A* decoding can be used to reduce the search space. If a state sequence is viewed as a path through a state-time lattice, at each point in the lattice only the path with the highest probability is selected.

The output of the acoustic modeling stage is a set of word hypotheses which can be examined to find the best word sequence, using a *language model* $P(W)$. The language model contains knowledge about which words are more likely in a given sequence. Two types of model are possible. A finite state network predicts all the possible word sequences in the language model. This approach is useful if all the phrases that are likely to occur in the speech input can be specified in advance. The disadvantage is that perfectly legal strings that were not anticipated are ruled out. Finite state networks can be used to parse well-defined sequences such as expressions of time.

Alternatively, an N-gram model can be used. The use of N-grams involves computing the probability of a sequence of words as a product of the probabilities of each word, assuming that the occurrence of each word is determined by the preceding $N - 1$ words. This relationship is expressed in the formula

$$\begin{aligned} P(W) &= P(w_1, \dots, w_n) \\ &= \prod_{n=1}^{N} P(w_n \mid w_1, \dots, w_{n-1}). \end{aligned}$$

However, because of the high computational cost involved in calculating the probability of a word given a large number of preceding words, N-grams are usually reduced to bigrams ($N = 2$) or trigrams ($N = 3$). Thus in a bigram model $P(w_i \mid w_{i-1})$ the probability of all possible next words is based only on the current word, while in a trigram model $P(w_i \mid w_{i-2}, w_{i-1})$ it is based on two preceding words. N-gram models may also be based on classes rather than words, i.e., the words are grouped into classes representing either syntactic categories such as noun or verb, or semantic categories, such as days of the week or names of airports. A language model reduces the *perplexity* of a system, which will usually result in greater recognition accuracy. Perplexity is roughly defined as the average branching factor, or average number of words, that might follow a given word. If the perplexity is low, recognition is likely to be more accurate as the search space is reduced.

The output of the speech recognizer may be a number of scored alternatives as in the following example representing the recognizer's best guesses for the input string *what time does the flight leave?* [Wyard et al. 1996]:

(1) what time does the white leaf 1245.6.
(2) what time does the flight leave 1250.1.
(3) what time does a flight leave 1252.3.
(4) what time did the flight leave 1270.1.
(5) what time did a flight leave 1272.3.

Sometimes there are only small differences between the alternatives, caused by one or two words that may not contribute to the meaning of the string. For this reason, the alternatives can be more economically represented in a directed graph or as a word lattice. The selection of the most likely sequence may be the responsibility of other system components. For example, if the domain of the dialogue system is flight enquiries, then the first sequence, which had the best score from the speech recognizer, would be discarded as contextually irrelevant. Similarly, dialogue information would assist the choice between 2 and 3, which ask about a flight departure that has not yet taken place, and 4 and 5, which ask about some departure that has already happened.

As an alternative to returning the complete sequence of words that matches the acoustic signal, the recognizer can search for key words. This technique is known as *word spotting*. Word spotting is useful for dealing with extraneous elements in the input, for example, detecting *yes* in the string *well, uh, yes, that's right*. The main difficulty for word spotting is to detect non-key-word speech. One method is to train the system with a variety of non-key-word examples, known as *sink (or garbage) models*. A word-spotting grammar network can then be specified that allows any sequence of sink models in combination with the key words to be recognized.

Users of spoken dialogue systems are generally constrained to having to wait until the system has completed its output before they can begin speaking. Once users are familiar with a system, they may wish to speed up the dialogue by interrupting the system. This is known as *barge-in*. The difficulty with simultaneous speech, which is common in human-human conversation, is that the incoming speech becomes corrupted with echo from the ongoing prompt, thus affecting the recognition. Various techniques are under development to facilitate barge-in.

4.1.1. Summary. This section has outlined the main characteristics of the speech recognition process, describing the uncertain and probabilistic nature of this process, in order to clarify the requirements that are put on the other system components. In a linear architecture the output of the speech recognizer provides the input to the language understanding module. Difficulties may arise for this component if the word sequence that is output does not constitute a legal sentence, as specified by the component's grammar. In any case, the design of the language understanding component needs to take account of the nature of the output from the speech recognition module. Similarly, in an architecture in which the dialogue management component interacts with each of the other components, one of the roles of dialogue management will be to monitor when the user's utterance has not been reliably recognized and to devise appropriate remedial steps. These issues will be discussed in greater detail in subsequent sections. For more extensive accounts of speech recognition, see, for example, Rabiner and Juang [1993] and Young and Bloothooft [1997]. For tutorial overviews, see Makhoul and Schwartz [1995] and Power [1996].

4.2. Language Understanding

The role of the language understanding component is to analyze the output of the speech recognition component and to derive a meaning representation that can be used by the dialogue control component. Language understanding involves syntactic analysis, to determine the constituent structure of the recognized string (i.e., how the words group together), and semantic analysis, to determine the meanings of the constituents. These two processes may be kept separate at the representational level in order to maintain generalizability to other domains, but they tend to be combined during processing for reasons of efficiency. On the other hand, some approaches to language understanding may involve little or no syntactic processing and derive a semantic representation directly from the recognized string. The advantages and disadvantages of these approaches, and the particular problems involved in the processing of spoken language, will be reviewed in this section.

The theoretical foundations for language processing are to be found in linguistics, psychology, and computational linguistics. Current grammatical formalisms in computational linguistics share a number of key characteristics, of which the main ingredient is a feature-based description of grammatical units, such as words, phrases and sentences [Uszkoreit and Zaenen 1996]. These feature-based formalisms are similar to those used in knowledge representation research and data type theory.

Feature terms are sets of attribute-value pairs in which the values can be atomic symbols or further feature terms. Feature terms belong to types, which may be organized in a type hierarchy or as disjunctive terms, functional constraints, or sets. The following simple example shows a feature-based representation for the words *lions*, *roar*, and *roars* as well as a simple grammar using the PATR-II formalism [Shieber 1986] that defines how the words can be combined in a well-formed sentence:

lexicon
lions: [cat:NP, head: [agreement: [number: plural, person:third]]]
roar: [cat:V, head: [form: finite, subject: [agreement: [number:plural, person:third]]]]
roars: [cat:V, head: [form: finite, subject: [agreement: [number:singular, person:third]]]]

grammar
S → NP VP
<S head> = <VP head>
<S head subject> = <NP head>

VP → V
<VP head> = <V head>

The lexicon consists of complex feature structures describing the syntactically relevant characteristics of the words, such as whether they are singular or plural. The grammar consists of phrase structure rules and equations that determine how the words can be combined.

The means by which feature terms may be combined to produce well-formed feature terms is through the process of unification. For example: the words *lions* and *roar* can be combined as their features unify, whereas *lions* and *roars* cannot, as the agreement features are incompatible. This basic formalism has been used to account for a wide range of syntactic phenomena and, in combination with unification, to provide a standard approach to sentence analysis using string-combining and information-combining operations.

Feature-based grammars are often subsumed under the term *unification grammars*. One major advantage of unification grammars is that they permit a declarative encoding of grammatical knowledge that is independent of any specific processing algorithm. A further advantage is that a similar formalism can be used for semantic representation, with the effect that the simultaneous use of syntactic and semantic constraints can improve the efficiency of the linguistic processing.

In computational semantics, sentences are analyzed on the basis of their constituent structure, under the assumption of the principle of compositionality, that is, that *the meaning of a sentence is a function of the meanings of its parts*. Each syntactic rule has a corresponding semantic rule and the analysis of the constituent structure of the sentence will lead to the semantic analysis of the sentence as the meanings of the individual constituents identified by the syntactic analysis are combined. The meaning representation from this form of semantic analysis is typically a logical formula in first order predicate calculus (FOPC) or some more powerful intermediate representation language such as Montague's intensional logic or Discourse Representation Theory (DRT). The advantage of a representation of the meaning of a sentence in a form such as a formula of FOPC is that it can be used to derive a set of valid inferences based on the inference rules of FOPC. For example, as Pulman [1996] showed, a query such as:

> *Does every flight from London to San Francisco stop over in Reykyavik?*

cannot be answered straightforwardly by a relational database that does not store propositions of the form *every X has property P*. Instead a logical inference has to be made from the meaning of the sentence based on the equivalence between *every X has property P* and *there is no X that does not have property P*. Based on this inference the system simply has to determine if a nonstopping flight can be found, in which case the answer is *no*; otherwise it is *yes*.

While linguistics and psychology provide a theoretical basis for computational linguistics, the characteristics of spoken language require additional (or even alternative) techniques. One problem is that naturally occurring text, both in written form, as in newspaper stories, as well as in spoken form, as in spoken dialogues, is far removed from the well-formed sentences that constitute the data for theoretical linguistics and psychology. In linguistics the main concern is with developing theories that can account for items of theoretical interest, often rare phenomena that demonstrate the wide coverage of the theory, while in psychology the main concern is with identifying the cognitive processes involved in language understanding. Traditionally a symbolic representation is used, with hand-crafted rules that produce a complete parsing of grammatically correct sentences but with a target coverage based on a relatively small set of exemplar sentences. When confronted with naturally occurring texts such as newspaper stories, these theoretically well-motivated grammars tend to generate a very large number of possible parses, due to ambiguous structures contained in the grammar rules, while, conversely, they often fail to produce the correct analysis of a given sentence, often having a failure rate of more than 60% [Marcus 1995].

Spoken language introduces a further problem in that the output from the speech recognizer will often not have the form of a grammatically well-formed string that can be parsed by a conventional language understanding system. Rather it is likely to contain features of spontaneous speech, such as sentence fragments, afterthoughts, self-corrections, slips of the tongue, or ungrammatical combinations. The following examples of utterances (cited in Moore [1995]), from a corpus collected from subjects using either a simulated or an actual spoken language Air Travel Information System (ATIS), would not be interpreted by a traditional linguistic grammar:

> *What kind of airplane goes from Philadelphia to San Francisco Monday stopping in Dallas in the afternoon (first class flight)*
>
> *(Do)(Do any of these flights)Are there any flights that arrive after five p.m.*

The first example is a well-formed sentence followed by an additional fragment or after-thought, enclosed in parentheses.

The second example is a self-correction in which the words intended for deletion are enclosed in parentheses.

Some of these performance phenomena occur sufficiently regularly that they could be described by special rules. For example, in some systems rules have been developed that can recognize and correct self-repairs in an utterance [Dowding et al, 1993; Heeman and Allen 1997]. A conventional grammar could be enhanced with additional rules that could handle some of these phenomena, but the problem is that it would be impossible to predict all the potential occurrences of these features of spontaneous speech in this way. An alternative approach is to develop more robust methods for processing spoken language.

Robust parsing aims to recover syntactic and semantic information from unrestricted text that contains features that are not accounted for in hand-crafted grammars. Robust parsing often involves partial parsing, in which the aim is not to perform a complete analysis of the text but to recover chunks, such as nonrecursive noun phrases, that can be used to extract the essential items of meaning in the text. Thus the aim is to achieve a broad coverage of a representative sample of language which represents a reasonable approximate solution to the analysis of the text [Abney 1997]. In some systems mixed approaches are used, such as first attempting to carry out a full linguistic analysis on the input and only resorting to robust techniques if this is unsuccessful. BBN's Delphi system [Stallard and Bobrow 1992], MIT's TINA system [Seneff 1992], and SRI International's Gemini system [Dowding et al. 1993] work in this way. As Moore [1995] reported, different results have been obtained. The SRI team found that a combination of detailed linguistic analysis and robust processing resulted in better performance than robust processing alone, while the best performing system at the same evaluation (the November 1992 ATIS evaluation) was the CMU Phoenix system, which uses only robust processing methods and does not attempt to account for every word in an utterance.

4.2.1. Integration of the Speech Recognition and Natural Language Understanding Components. So far it has been assumed that the speech recognizer and the natural language understanding module are connected serially and that the speech module outputs a single string to be analyzed by the language understanding module. Typically, however, the output from the speech recognition component is a set of ranked hypotheses, of which only a few will make sense when subjected to syntactic and semantic analysis. The most likely hypothesis may turn out not to be the string that is ranked as the best set of words identified by the speech recognition component (see the example in Section 4.1). What this implies is that, in addition to interpreting the string (or strings) output by the speech recognizer to provide a semantic interpretation, the language understanding module can provide an additional knowledge source to constrain the output of the speech recognizer. This in turn has implications for the system architecture, in particular for the ways in which the speech recognition and natural language understanding components can be linked or integrated.

The standard approach to integration involves selecting as a preferred hypothesis the string with the highest recognition score that can be processed by the natural language component. The disadvantage of this approach is that strings may be rejected as unparsable that nevertheless represent what the speaker had actually said. In this case the recognizer would be overconstrained by the language component. Alternatively, if robust parsing were applied, the recognizer could be underconstrained, as a robust parser will attempt to make sense out of almost any word string.

One alternative approach to integration is word lattice parsing, in which the recognizer produces a set of scored word hypotheses and the natural language module attempts to find a grammatical utterance spanning the input signal that has the highest acoustic score. This approach becomes unacceptable in the case of word lattices containing large numbers of hypotheses, particularly when there is a large degree of word boundary uncertainty. Another alternative is to use N-best filtering in which the recognizer outputs the n-best hypotheses (where N may range from between 10 to 100 sentence hypotheses), and these are then ranked by the language understanding component to determine the best-scoring hypothesis [Price 1996]. This approach has the advantage of simplicity but the disadvantage of a high computational cost given a large value for N. Many practical systems have, however, produced acceptable results with values as low as $N = 5$, using robust processing if strict grammatical parsing was not successful with the top five recognition hypotheses [Kubala et al. 1992].

4.2.2. Some Solutions. Various solutions have been adopted to the problem of deriving a semantic representation from the string provided by the speech recognition component. These include: comprehensive linguistic analysis, methods for dealing with ill-formed and incomplete input, and methods involving concept spotting. Some of these will be briefly reviewed in the following paragraphs.

4.2.2.1. SUNDIAL. In the SUNDIAL project [Peckham 1993], which was concerned with travel information in English, French, German, and Italian, several different approaches were adopted, with the following common features:

- —a rich linguistic analysis;
- —robust methods for handling partial and ill-formed input;
- —a semantic representation language for task-oriented dialogues.

Linguistic analysis in the German version is based on a chart parser using a unification categorial grammar [Eckert and Niemann 1994]. Syntactic and semantic structures are built in parallel by unifying complex feature structures during parsing. The aim is to find a consistent maximal edge of the utterance, but if no single edge can be found, the best interpretation is selected for the partial descriptions returned by the chart parser. These partial descriptions are referred to as *utterance field objects* (UFOs). Various scoring measures are applied to the chart edges to determine the best interpretation. Additionally some features of spontaneous speech such as pauses, filled pauses, and ellipses, are represented explicitly in the grammar. The following example illustrates the use of UFOs in the analysis of the string *I want to go—at nine o'clock from Koeln* [Eckert and Niemann 1994]:

U1: syntax: [string: 'I want to go']
semantics: [type:want, theagent: [type: speaker], thetheme: [type: go]]
U2: syntax: [string: 'at nine o'clock']
semantics: [type: time, thehour: 9]
U3: syntax: [string: 'from Koeln']
semantics: [type: go, thesource: [type: location, thecity: koeln]]

This sequence of UFOs is a set of partial descriptions that cannot be combined into a longer spanning edge, as U2, an elliptical construction, is not compatible with U1 and U3. However, it is still possible to build a semantic representation from these partial descriptions, as shown in this example.

This example also illustrates the *semantic interface language* (SIL) which is used in SUNDIAL to pass the content of messages between modules. Two different levels of detail are provided in SIL, both in terms of typed feature structures: a linguistically oriented level, as shown above, and a task-oriented level, which contains information relevant to an application, such as relations in the application database. The task-oriented representation for the partial descriptions in the example above would be:

U1: [task param]: [none]]
U2: [task param]: [sourcetime: 900]]
U1: [task param]: [sourcecity: koeln]]

This task-oriented representation is used by the dialogue manager to determine whether the information elicited from the user is sufficient to permit database access or whether further parameters need to be elicited.

Reporting on a comparative evaluation between earlier versions of the system, which did not include a robust semantic analysis, and a later version that did, Eckert and Niemann [1994] found a much better dialogue completion rate in the later system, even though word accuracy rate (the results from the speech recognizer) had remained roughly constant across the systems.

4.2.2.2. SpeechActs. The SpeechActs system [Martin et al. 1996], which enables professionals on the move to obtain on-line information and services, uses a feature-based approach to encode the semantic content of utterances in terms of the underlying application that is being accessed, such as the Mail or Calendar tool. The analysis aims to provide an accurate understanding of the input while tolerating misrecognized words. Thus the analysis is more comprehensive than keyword matching, which would miss subtle shades of meaning, and a full semantic analysis, which might fail due to missing or misrecognized words from the speech recognizer. Because there is wide variation in the linguistic demands of the different applications, the system is reset for a new lexicon and grammar for each new application.

Generally a grammar for a speech recognizer differs from the grammar of a language understanding component. The function of a speech recognition grammar is to determine the words that were spoken and to specify every possible utterance that a user might say to the system. For this purpose a finite state grammar or a language model is normally used, as described earlier. The role of the language understanding component is to extract the meanings of the words, and for this purpose a more general grammar is required. Two problems arise, however. First, given that the grammar formalisms for different recognizers vary widely, a developer would have to write a different version of the recognizer grammar each time a new recognition system was used.[2] A second problem concerns the degree of integration between the speech recognition and language understanding components. With different grammar types this integration would be less feasible.

The solution that was adopted was a Unified Grammar, which could be compiled into a speech recognizer grammar that would include constraints to help reduce perplexity, as well as into a corresponding grammar for natural language processing [Martin et al. 1996]. A Unified Grammar consists of a collection of rules that include finite state patterns and augmentations. An example of a pattern is

"what" root="be" ("in"–"on") namePossessive sem=calendar;

This pattern requires the first word to be *what*, then a form of the verb *be*, then either *in* or *on*, followed by the output of the rule *namePossessive* (for example: *Paul's*), and finally a word with the semantics of "calendar." Augmentations include tests such as

be.past-participle ! = t; be.ing-form ! = t;

stating that the form of the verb must not be *been* or *being*, and the action

action := 'lookup';

which adds the feature "lookup" to the action associated with this pattern. One advantage of these grammar rules for the language understanding process is that a number of different utterances with the same meaning will result in the same analysis. A second advantage is flexibility, which is achieved through the use of wild cards in the patterns to cope with confusions arising from the speech recognizer between insignificant words such as *of*, *a*, or *the*, which, if misrecognized, would pose problems for a conventional language processor.

4.2.2.3. The Philips Automatic Train Timetable Information System. A rather different approach was adopted in the Philips train timetable information dialogue system [Aust et al. 1995], which was illustrated in Section 3. This system also accepts partial structures as input, similar

---

[2] Note: due to rapid changes in technology, the developers of SpeechActs did not want to restrict future developers to specific speech recognizers but to allow them to use newly available technologies as plug-in components.

to those accepted by the SUNDIAL and SpeechActs systems. The speech recognition module generates word graphs, consisting of about 10 edges per graph after graph optimization. Each path through the graph represents a sentence hypothesis. The language understanding module has to find the best path through the graph and then determine its meaning. However, language analysis in this system involves a search for meaningful concepts using an attributed context-free stochastic grammar to identify the relevant phrases and compute their meanings. Thus the grammar is semantic rather than syntactic since it is not concerned with the structure of the sentence but with its meaning. In addition, each rule may have a probability indicating how likely it is to be applied, given its left hand-side nonterminal. The following is an example of some grammar rules for a train timetable application:

```
<station> STRING
          : 'London'                  {'London';}
          | 'Paris'                   {'Paris';}
          ;
<origin>  STRING
          : 'from' <station>          {#2;}
          | 'not' 'from' <station>    {NOT #3;}
          ;
```

Rules have a syntactic part in which the left-hand side—preceding the colon—comprises a nonterminal (in angular brackets) and the right-hand side comprises a sequence of nonterminals and/or terminals (enclosed in single quotes). The semantic rules, which contain assignments and expressions, are enclosed in braces. The semantic part may also contain a speech understanding expression represented as an attribute reference, with a number following the # symbol indicating the position of the nonterminal within an assigned sequence. Links are provided between the attributes of the grammar rules and the variables (or slots) that represent the concepts for which values are to be obtained during the dialogue.

As many of the input strings are incomplete or ungrammatical, in terms of a traditional sentence-based grammar, the system can generally still derive a meaning for these strings at a fairly low computational cost. This involves procedures for dealing with filler arcs, that is, those parts of a graph that do not contain identified concepts (what are described as *meaningful fillers*). In addition to this, a concept bigram model is used to model more frequently observed concept sequences.

A semantics-driven approach such as this permits the analysis of the ungrammatical strings that characterize naturally occurring spoken language. For example: given an input string such as

> *I want to uh let me see from Frankfurt yes is there a train to Hamburg from Frankfurt at about 10 o'clock?*

the system could identify the essential concepts such as *source (from + Placename)*, *destination (to + Placename), time (at + time expression)* and compute a meaning for the sentence, ignoring the meaningful fillers in the string (*I want to, let me see, yes is there a train*, etc.). However, as Moore [1995] pointed out, concept spotting might not be able to handle more complex examples such as

> *What cities does the train from Hamburg stop at?*

as the case marking word *at* is separated from its associated case word *what cities*. Only a more sophisticated grammatical analysis could determine this sort of relationship between disjoint constituents. Generally speaking, this type of construction is relatively infrequent and, if encountered sufficiently frequently, could be modeled in the grammar. In any case, a system with adequate repair facilities should be capable of eliciting the required information from the user through more directed questions and prompts. Some of these techniques for achieving this type of robust behavior will be described below.

4.2.3. Summary. This section has examined the role of the language understanding component and issues such as grammar representation and robust parsing. The construction of the language understanding component is determined on the

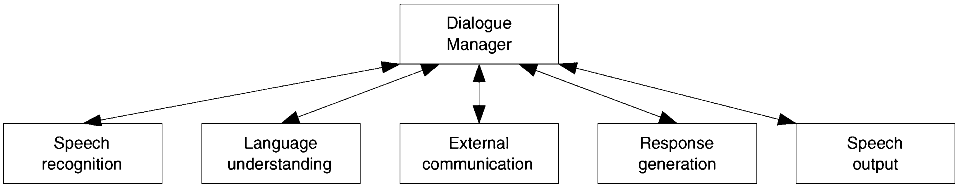


Fig. 7. An architecture for spoken dialogue systems.

one hand by the nature of the input as received from the speech recognizer, and, on the other hand, by the type of input required by the dialogue manager. Many systems, including the ATIS systems and the Philips train timetable information system, are essentially driven by a template-filling mechanism. In the ATIS systems the dialogue is driven by the user who submits queries to the system, while the system only takes the initiative to elicit missing information in the template. The Philips system is system-led, with the selection of questions determined by the system and the next step in the dialogue being determined by what is missing or needs to be clarified or confirmed in the template. The SUNDIAL system is similarly driven by the goal of eliciting the information required to fill a template and execute a database query, but it is based on a more complex architecture for dialogue management that requires more extensive processing of the linguistic input. The SpeechActs language understanding component is carefully engineered in terms of the underlying application, with the main aim being to provide a semantic analysis that can be handled by the system's dialogue manager in spite of speech recognition errors. How a dialogue manager handles the input from the language understanding component and generates output to the user will be examined in the next section.

4.3. Dialogue Management

The main function of the dialogue management component is to control the flow of the dialogue. This involves the following tasks:

—determining whether sufficient information has been elicited from the user in order to enable communication with the external application;
—communicating with the external application;
—communicating information back to the user.

In a simple architecture these tasks could be seen in terms of a serially ordered set of processes: the system finds out what the user wants to know or do, consults the external application, and reports the results back to the user. More typically, however, the process of determining what the user wants to know or do will be more problematic, as the information elicited from the user may not be sufficient to enable the system to consult the external application, for reasons such as the following:

—the user's input may be ill-formed, with the result that it could not be sufficiently interpreted by the speech recognition and language understanding components;
—the user's input may be incomplete or inaccurate, with the result that insufficient information is available to consult the external application.

This subsection will describe the issues involved in dealing with ill-formed, incomplete, or inaccurate input from the user. Methods for controlling the dialogue flow will be discussed in Section 5.

Various error-handling processes are required to deal with these situations, involving clarification and verification subdialogues between the system and the user. This implies a more complex and more integrated system architecture, in which the dialogue management component has a central controlling function, as shown in Figure 7. Variations on this architecture will be discussed in greater detail in Section 5. The next two subsections will examine two important functions of the dialogue manager:

—dealing with input that the system recognizes as ill-formed or incomplete;

—using confirmation strategies to verify that the input recognized by the system is indeed what was intended by the user.

4.3.1. Dealing with Ill-Formed or Incomplete Input.

The simplest way of dealing with ill-formed or incomplete input is to simply report the problem back to the user and to request a reformulation of the input. This method is clearly inadequate as it fails to distinguish the different ways in which the input is ill-formed or incomplete, and it fails to support the user in reformulating the input. A number of different solutions have been proposed.

Assuming that the user's input has been processed by the speech recognition and language understanding components using the methods described in Sections 4.1 and 4.2, a number of higher level knowledge sources can be brought to bear to assist in the interpretation of ill-formed input. The main knowledge sources that have been used for this purpose include interpretation based on

—speech acts;

—the discourse context;

—the dialogue structure;

—a user model.

4.3.1.1. Interpretation based on speech acts.

The concept of speech acts emerged out of the work of Austin [1962] in the philosophy of language and was further developed by Searle [1969]. A speech act is defined as the function of an utterance—for example, a request, promise, warning, or piece of information. Speech act analysis involves a higher level of analysis than syntactic and semantic analysis, requiring reference to external information such as the discourse context and the speaker's beliefs and desires. For example: the utterance *It's cold in here* has the syntactic form of a declarative and a literal semantic reading describing the level of temperature. However, while in a particular context the utterance might have this literal reading, in a different context it could function as a request (for example, to close a window or turn on the heating). Speech act theory has been used as a basis for computational theories of communication in which responses to a speaker's utterances are guided by the hearer's recognition of the intentions underlying the utterances [Allen and Perrault 1980; Cohen and Levesque 1990].

Speech act analysis has been used in the TRAINS project to support the interpretation of ill-formed input. The TRAINS project [Allen et al. 1995] is concerned with the development of dialogue technology in support of collaborative problem solving. The current work has involved an interactive planning assistant that engages in dialogue with a user to solve route-planning problems. The linguistic analysis of the user's utterances is constructed by a bottom-up parser with a feature-based grammar, in which each rule specifies syntactic and semantic constraints. However, the output of the parser is not a syntactic or semantic analysis, but rather a sequence of speech acts that provide the "minimal covering" of the input, that is, the shortest sequence that accounts for the input. This enables the parser to output an analysis even if an utterance is otherwise uninterpretable [Allen et al. 1996]. For example: the utterance *Okay now let's take the last train and go from Albany to Milwaukee* was output from the speech recognizer as *okay now I take the last train in go from albany to is*. The best sequence of speech acts that covered this input was

(1) a CONFIRMATION/ACKNOWLEDGE (*okay*);

(2) a TELL, with content to take the last train (*now I take the last train*);

(3) a REQUEST to go from Albany (*go from albany*).

Although the system was unable to perform a complete syntactic and semantic analysis of this utterance, enough information was extracted to enable the system to continue the dialogue, in this case by generating a clarification subdialogue. Thus robust parsing using speech acts enabled the system to compensate for the errors emanating from the speech recognition component.

4.3.1.2. Interpretation based on the discourse context. Discourse context is often used to assist with the interpretation of items that are otherwise uninterpretable out of context, such as pronouns (*he*, *she*, *they*, etc.) and deictic expressions such as *this one*, *the next one*, *the previous flight*. These expressions, which usually refer to some item that has been mentioned previously in the dialogue, can only be interpreted if some record of previously mentioned items has been kept. Similarly, it is often not possible to interpret input that is incomplete due to ellipsis. Ellipsis involves clauses that are syntactically incomplete in which the missing parts can be recovered from a previous main clause—for example, where in response to the question *Which flight do you wish to book?* the user says *The London one*. In this case the incomplete input has to be interpreted in terms of the preceding question, that is, *I wish to book the London flight*.

Considerable attention has been devoted in computational theories of discourse to the issues associated with discourse context. The simplest method involves maintaining a history list of elements mentioned in the preceding discourse that can be referred to using pronouns. A more elaborate approach involves the concept of *centering*, which states a preference for pronominalization of entities in the history list that play a central role in a main clause over those in subordinate and adjunct clauses. In other words, certain entities, such as those in the subject position in a main clause, are the center of the discourse focus and can be referred to using pronouns in the next few sentences until the focus shifts to another entity. Grosz, et al. [1983] introduced the concept of centering to computational discourse analysis, while Walker [1989] has compared the simpler use of history lists with analysis based on centering.

More generally, discourse phenomena such as anaphora and ellipsis require some representation of the local context that contains the syntactic and semantic structures of previous clauses. Sophisticated language understanding systems normally include such a discourse-processing component, sometimes referred to as the *pragmatics module*, to deal with issues of context.

4.3.1.3. Interpretation based on the dialogue structure. Interpretation based on the dialogue structure makes use of the expectations provided by the dialogue model. The essential idea is that at each point in a dialogue there are constraints on what can be said next. These constraints can be of various types and can assist several components of a dialogue system. The constraint that the next user utterance is likely to be some form of the words *yes* or *no*, because the system prompt was in the form of a *yes/no question*, constrains the recognizer to having to deal with only this limited vocabulary at this point in the dialogue. Similar expectation-based constraints can help determine which grammatical and semantic rules need to be active at any given point in the dialogue.

In contrast to the methods described for the resolution of anaphora and ellipsis, which are usually only applied after a sentence has been completely recognized by the speech recognition component, Young et al. [1989] in the MINDS-II system proposed the use of higher-level knowledge sources to assist speech recognition by reducing the search space for the words in the speech signal. Unlike the language models described earlier, this approach involves the dynamic prediction of the concepts that are likely to be referred to in the user's next utterance, based on the previous user query, the database response, and the current state of the dialogue. The system predicts which goals and subgoals the user is likely to try to complete at the next point in the dialogue. These are combined with information from a user model that represents the domain concepts and relations among concepts that the user is likely to know about. The predicted concepts are then translated into word sequences that denote these concepts and these word sequences are combined into a semantic network that represents a maximally constrained grammar at this particular point in the dialogue. The results indicated that the use of these higher-level knowledge sources reduced test set perplexity from 279.2 for a complete grammar to 17.8 when best predictions were used, while the speech recognition error rate decreased from 17.9% to 3.5%.

Expectation-driven processing is also used in the Circuit-Fix-It Shop system to assist with the interpretation of the user's utterances. The approach used is based on the notion of *attentional state* as described in the theory of discourse structure of Grosz and Sidner [1986]. Essentially the attentional state refers to the focus of attention of the conversational participants. In this system the key idea is that at any point in the dialogue there is a particular task step which is under discussion and that, given this information, the system can derive a set of expectations of what the user will say next. To take a particular example [Smith and Hipp 1994]:

> Following a statement by the system that is an attempt to have a specific task step S performed, where S and another task step T must be completed as part of the performance of task step R, there are expectations for any of the following types of response:
>
> (1) a statement about missing or uncertain background knowledge necessary for the accomplishment of S (e.g. *how do I do substep* S_1*?*)
> (2) a statement about a subgoal of S (e.g. *I have completed substep* S_1)
> (3) a statement about the underlying purpose for S (e.g. *why does S need to be done?*)
> (4) a statement about ancestor task steps of which accomplishment of S is a part (e.g. *I have completed R*)
> (5) a statement about another task step which, along with S, is needed to accomplish some ancestor task step (e.g. *how do I do T?*)
> (6) a statement indicating accomplishment of S (e.g. *S is done*)

These expectations are computed from two sources:

(1) the domain processor, which provides situation-specific expectations based on the actions applicable within the domain;
(2) the dialogue controller, which has knowledge about the general nature of task-oriented dialogues.

For example: if the topic is *What is the LED displaying?* then the expectations provided by the domain processor would consist of the possible descriptions of the LED display. With the same topic, the dialogue controller would examine the input at a more abstract level as a query about the *observing* of the *value* for a *property* of an *object*. Expectations about observing *property* values of an *object* would include questions about its location, how to perform the observation, or definitions of the *property*.

The expectations that are computed are used to provide several types of contextual interpretation of the user's utterances, including the resolution of anaphoric reference and elliptical responses as well as the maintenance of dialogue coherence when dealing with clarification subdialogues. The expectations also assist the language understanding component, as in the MINDS-II system, by providing a set of strings that the dialogue controller expects the user to say next, along with an estimate of the probability of each string.

4.3.1.4. Interpretation based on a user model. Information about the user, often referred to as a *user model*, is a further source of information to assist with the interpretation of the user's utterances. User modeling first emerged in the context of the natural language dialogue systems of the early 1980s as a means of providing more cooperative conversational behavior through the use of a model of the user's beliefs, goals, and plans [Wahlster and Kobsa 1989]. One aspect of cooperative conversational behavior that was investigated is the ability to respond to a user's queries that are underspecified or ill-formed by inferring the plan underlying the user's query and responding in terms of the plan. Carberry [1989] developed a system for incrementally constructing a model of the user's plans from an ongoing dialogue and then using this model to interpret subsequent utterances. At the initial point in the dialogue the system has no prior context to consider. The utterance is interpreted in terms of a set of domain-dependent plans and a set of candidate goals is derived that are incorporated into one or more models of the user's plan that can be used to provide a context for the interpretation of future utterances. As further utterances are input, these are matched with the current context models, which may either be expanded to incorporate a further aspect of the current focused plan, or may be discarded as no longer relevant. User models were also employed in other work by Carberry and others to handle elliptical queries and pragmatically ill-formed queries—that is, queries involving misconceptions, where the user's beliefs differ from those of the system (see also Pollack [1986]). The use of information about the user's beliefs, goals, and intentions in more recent dialogue systems such as TRAINS will be reviewed in Section 5 in terms of BDI (Belief, Desire, and Intention) architectures.

4.3.2. Confirmations and Verifications. The various knowledge sources described in Section 4.3.1 enable the dialogue system to compensate for input that is ill-formed or incomplete without having to consult the user with requests for repetition or clarification. On the other hand, verification is required to deal with potentially misrecognized input where the system "realizes" that it may have misrecognized or misunderstood what the user said. Verification is common in human-human dialogues to ensure that the information conveyed is mutually understood and that a common ground is established and maintained [Clark 1992]. Confirming that the system has understood what the user actually intended is even more necessary in spoken dialogues with computers given the possibility of recognition and understanding errors. There are several different ways in which a dialogue system can verify or confirm that the user's utterances have been correctly understood. Some examples (from [Gerbino and Danieli 1993]) are provided to illustrate these different types.

4.3.2.1. Explicit Verification. Explicit verification takes the form of a question that asks explicitly for confirmation of the input. This may be accompanied by a request to answer with *yes* or *no*:

> *Do you want to go from Trento to Milano? Yes or no?*

In this case two values are confirmed at the same time. If one value is incorrect, the user is able to correct, (e.g., *No to Torino*), although it is problematic if both values are incorrect as the potential for misrecognition is increased if the user has to correct both values (e.g., *No from Torino to Merano*). An alternative is to confirm each value separately:

> *Do you want to go from Trento? Do you want to go to Milano?*

Although this is a more robust method for confirming values, the disadvantage is that the number of turns required to complete the dialogue is increased.

4.3.2.2. Implicit verification. With implicit verification the system embeds in its next question a repetition of its understanding of what the user said. The user can still correct the repeated value, but if the user answers the question without correcting the value, then that value has been implicitly confirmed:

User: *I want to travel from Milano to Roma.*
System: *At what time do you want to leave from Merano to Roma?*
User: *No I want to leave from Milano in the evening.*

In this case the user rejects the departure city proposed by the system and offers a correction, at the same time answering the question about the departure time. This strategy is closer to natural conversation and can be employed if the system is reasonably confident with the output from the speech recognition and language understanding components. The main problem is that implicit requests for verification can give rise to a wider range of possible responses, which may subsequently put greater demands on the recognition and understanding processes through increased complexity. A further problem is that verification requests involve a combination of the particular values that were misrecognized and those new values that are to be elicited in the system's question. With even a relatively small number of values the combinatorial possibilities expand rapidly, so that the questions cannot be determined in advance and a specific language generation component is required to generate the questions on the fly.

Verification may take place over a series of utterances that involve the application of a sequence of different types of question to ensure a correct response. The following example illustrates an explicit recovery strategy [Gerbino and Danieli 1993]:

User1: *I want to go to Milano leaving from Torino early in the morning.* (Recognition problems—EMPTY INPUT)
System1: *I haven't understood. Please rephrase your sentence.*
User2: *I want to go from Torino to Milano and I want to leave early in the morning.* (Recognition problems—EMPTY INPUT)
System2: *I haven't understood. Where do you want to leave from?*
User3: *From Torino, early in the morning.* (Recognition problems—EMPTY INPUT)
System3: *I haven't understood. Tell me the name of the departure city, only one word.*
User4: *Torino.*
System3: *Do you want to leave from Torino?.*
User4: *Yes.*

At User1 the Dialogue Manager receives empty input from the speech recognizer and asks for a rephrase (System1). After the next user utterance, the input is still empty, so the Dialogue Manager starts a sequence of requests to collect the parameters required to query the data base (System2). The Dialogue Manager uses predictions—expecting a departure city—to interpret the user's next input (User3). As this strategy also fails, the Dialogue Manager switches to isolated word recognition (System3). Following recognition of the input, the Dialogue Manager confirms the parameters before continuing (System4). The interaction style can now switch back to continuous speech. The following example illustrates an alternative implicit recovery strategy [Gerbino and Danieli 1993]:

User1: *I want to go to Milano leaving from Torino early in the morning.* (Recognition problems)
System1: *Sorry, where do you want to leave from?*
User2: *From Torino, early in the morning.*
System2: *Do you want to go from Torino to Milano early in the morning?*
User3: *Yes.*

When the Dialogue Manager receives ill-formed input (either fewer parameters than expected, or some unexpected parameters), it uses predictions based on the current point in the dialogue to interpret the input, that is, it expects a departure city (System1–User2). The user inputs the departure time as well as the departure city (User3). Here the departure time can be ignored as it was already introduced in User1. All the parameters introduced in User1 (departure time, arrival city) are stored and used in the confirmation in System2, while at the same time asking for the missing parameter (departure city).

Verification is one of the most challenging issues in spoken dialogue systems. A failure to verify correctly may lead to miscommunication, while an explicit verification strategy may result in an unreasonably lengthy dialogue, which has an adverse effect on user satisfaction. For example: in one experiment in which a

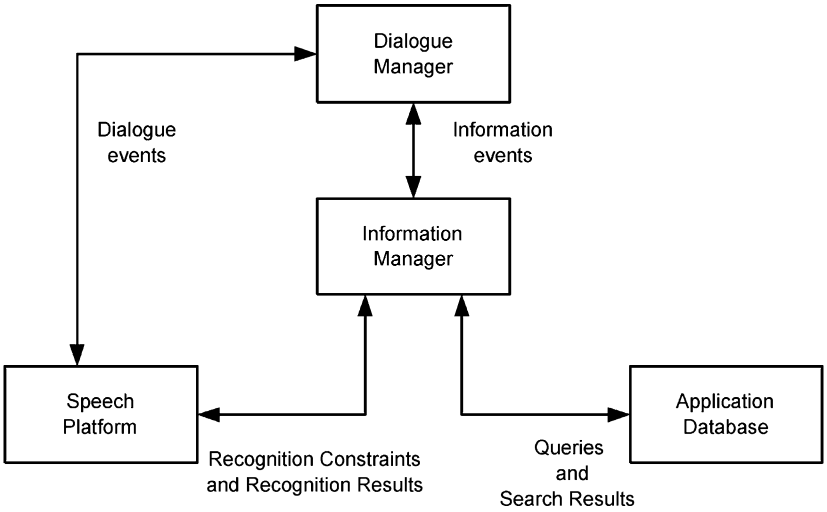


Fig. 8. An architecture including an Information Manager.

user had to read a credit card number consisting of four blocks of four digits, it was reported that the worst case involved 100 spoken prompts before the dialogue was successfully completed [Vergeynst et al. 1993]. Considerable research is being directed toward the development of effective and efficient verification strategies that allow the system to degrade gracefully, for example, by moving down a hierarchy from implicit to explicit verification, and finally to spelling mode, to elicit the required value, and then switching back to higher-level modes once the problem has been resolved. Applications of these and similar strategies have been reported in papers on the SUNDIAL project [Heisterkamp and McGlashan 1996] (see also Section 5.3.4), as well as in an AAAI workshop on miscommunication in human-machine dialogue [McRoy 1996].

4.4. External Communication

Generally dialogue systems require some form of communication with an outside source such as a database in order to retrieve the information requested during the course of the dialogue. Most dialogue systems communicate with a database. For example: in the Philips train timetable information system the user supplies the required parameters, such as source and destination stations, date, and time of departure or arrival, that enable the system to execute a database query. In other cases, the external communication may be with a knowledge base or with a planning system. Each of these possibilities will be examined in the following subsections.

4.4.1. Communicating with a Database. The processes involved in communicating with a database are not generally discussed in the spoken dialogue systems literature, presumably on the assumption that once the parameters of the query have been elicited during the course of the dialogue, accessing the database to retrieve the required information is a straightforward process. However, problems may arise if the vocabulary of the dialogue does not map directly on to the vocabulary of the application, if the query makes false assumptions concerning the actual contents of the database so that no straightforward response is possible, or if the data that is retrieved is ambiguous or indeterminate.

One solution to the problem of mapping between the vocabularies of the application and of the dialogue is to add an additional component to the architecture as shown in Figure 8. This is the approach adopted by Whittaker and Attwater [1996], who separated the dialogue and information aspects of the system and assigned any complex information processing that is required to an Information Manager. The Information Manager interacts on the one hand with the Dialogue Manager to communicate information events, and on the other hand with the application database to handle queries and search results. An additional link in the architecture shown in Figure 8 is to the Speech Platform to provide recognition constraints through the generation and manipulation of speech recognition vocabularies. Variations on this architecture will be discussed in greater detail in Section 5.

Some of the functions of the Information Manager may include resolving items of information that can be expressed in several ways, such as different spellings for surnames, shortened forms of first names, and problems associated with place names. More generally, the Information Manager may be responsible for relationships within vocabularies, such as synonyms and homophones, the manipulation of database hypotheses, such as scoring of partial and complete entries, and interactions with the application database. The key structure is the data model, which contains a number of vocabulary models that are each associated with one vocabulary within the application, so that a distinction can be made between how items are represented in the database and how they may be referenced within a spoken dialogue.

Database queries may be ill-formed because the user has misconceptions about the contents of the database. This issue has received wide attention in the field of natural language interfaces to databases, but has not yet been incorporated into spoken dialogue systems, due to the need to deal with more basic problems resulting from speech recognition and language processing errors. Early work by Kaplan [1983] addressed the issue of false assumptions, as illustrated in the following example:

User: *How many students got As in Linguistics in 1991?*
User: *None.*

The system's response is correct if the set of students that got an A in Linguistics is empty, but it would also be correct if there were no students taking linguistics in 1991. However, in the latter case the system's response is misleading, as it does not correct the user's false assumptions. Kaplan proposed solutions to this problem using *corrective indirect responses* and *suggestive indirect responses*.

Problems may also arise if the user has misconceptions about the world model represented in the database. Carberry [1986] discussed the query *Which apartments are for sale?* which (in an American real-estate context) is inappropriate, as apartments are rented, not sold, although apartment blocks may be sold, for example, to property developers. Resolving this problem involved discerning the user's goal that gave rise to the ill-formed query. Other approaches involve identifying objects and their attributes that have been incorrectly referenced and substituting a viable alternative [McCoy 1986].

Problems of ambiguous or indeterminate data have been treated to some extent in spoken dialogue systems, usually with some mechanism that has been specifically devised to handle problems that have been predicted in advance. For example, the Philips Dialogue Description Language [Philips Speech Processing 1997] has a mechanism for handling underspecified or ambiguous values, such as more than one train station with the same name. There are also mechanisms for combining values, for example, if a user calls in the afternoon with the utterance *Today at 8*, the two values can be combined into the single value 20.00 hours. Similarly, values that contradict, such as the same value for a destination and a source, can be detected.

Database access may be unsuccessful because a value did not find an exact match in the database. For example: a query concerning a flight to London at 8 might be unsuccessful, although there may be flights to London just before or just after this time. Enabling such a query in the SUNDIAL system involved relaxing some of the parameters of the query

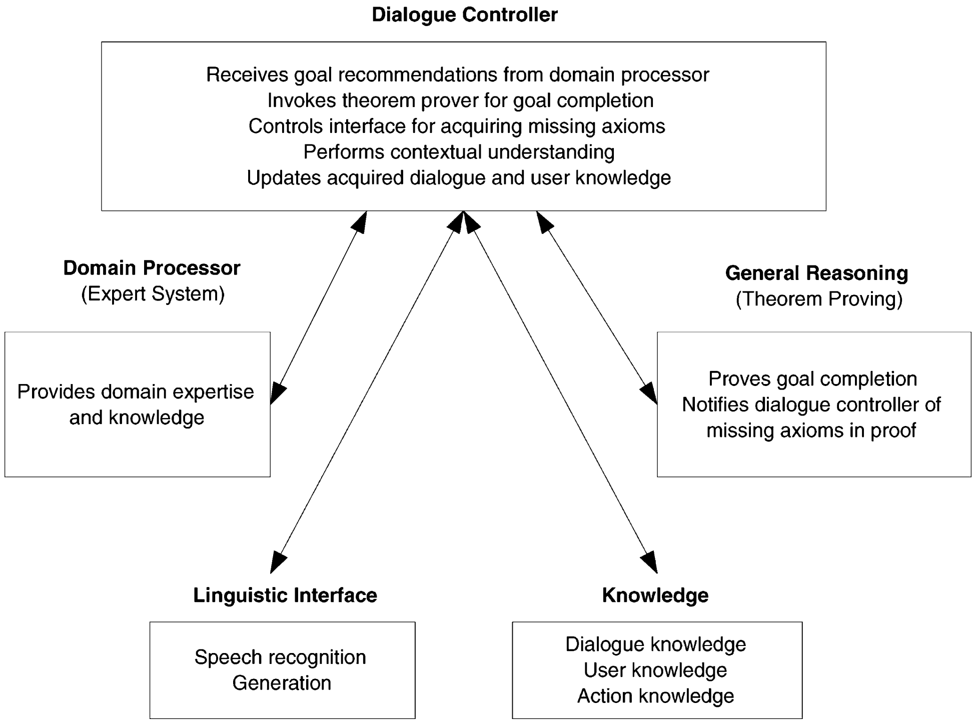

Dialogue Controller
Receives goal recommendations from domain processor
Invokes theorem prover for goal completion
Controls interface for acquiring missing axioms
Performs contextual understanding
Updates acquired dialogue and user knowledge
Domain Processor
(Expert System)
Provides domain expertise and knowledge
General Reasoning
(Theorem Proving)
Proves goal completion
Notifies dialogue controller of missing axioms in proof
Linguistic Interface
Speech recognition
Generation
Knowledge
Dialogue knowledge
User knowledge
Action knowledge

Fig. 9. The architecture of the Circuit-fix-it-Shop system.

[Giachin and McGlashan 1997]. In this case, the destination would not be an obvious candidate for relaxation, as the user would probably not want a flight to some other destination that leaves at 8. Relaxing the time would involve a simple relaxation algorithm that computes time intervals at increasing distances (for example, 5 minutes per iteration) from before and after the requested time, up to a given threshold.

Finally, there are problems concerning how the output is to be presented to the user. If a number of database solutions have been found, it is necessary to decide how much to present. In SUNDIAL the interval of solutions was divided into four subintervals:

$0 \ldots \text{MinGoal} \ldots \text{MaxGoal} \ldots \text{Threshold} \ldots,$

where MinGoal and MaxGoal represented the optimum range of solutions that were presented directly, and entries between MaxGoal and Threshold were presented only in summary form [Giachin and McGlashan 1997]. Some of these issues are dealt with in more detail in Section 4.5 (response generation) and Section 4.6 (speech output) of this article.

4.4.2. Communicating with a Knowledge Base. Communication with a knowledge base is required for systems that support problem solving rather than information retrieval. The Circuit-Fix-It Shop system [Smith and Hipp 1994], introduced in Section 3.3, which helps users to fix an electronic circuit, is a good example of such a system. In addition to the usual components that have been described so far, this system also includes the types of component found in knowledge-based systems, such as a *domain processor*, a *general reasoning* module, and a *general knowledge* component. The relationships between these components and the linguistic and dialogue processing components of the system are shown in Figure 9 [Smith and Hipp 1994]. The *domain processor* is the application-dependent component of the system—dealing in this case with electronic circuit repair. This component contains all the information about the application domain that enables it to recommend to the dialogue controller the steps that are required to accomplish a task using a special notation GADL (Goal Action and Description Language) to represent goals, actions and states. The domain processor also receives information back from the dialogue controller, which will have communicated with the user to obtain information about the current actions and states. This in turn enables the domain processor to update its world model and then to propose the next task step to be achieved.

While the domain processor is application-specific and in principle can be substituted by any other application domain, the *general reasoning* component is domain-independent and incorporates the general mechanisms for reasoning with the knowledge contained in the domain processor. In the case of the Circuit-Fix-It Shop system this reasoning takes the form of an interruptible theorem prover that requires interaction with the outside world to resolve missing axioms. This component provides the basis for the dialogue control mechanisms of the Circuit-Fix-It Shop system and will be discussed in greater detail in Section 5.

The *knowledge* component, which is also application-independent, includes knowledge relevant to task-oriented dialogues, such as how actions decompose into subactions and how theorems are used to prove goal completion. Among the other types of knowledge included in this component are general dialogue knowledge about the linguistic realisations of task expectations and knowledge about the user that is acquired during the course of the dialogue.

Problem solving in the Circuit-Fix-It Shop system involves cooperating with the user to solve a specific goal, such as how to repair a particular circuit. Problem solving is achieved through communication between the system and the user to establish what actions have to be carried out and what the current state of the task might be. The components described in this section support the system in its reasoning about the steps required to complete a task, in deciding what information to communicate to the user, and in the integration of information provided by the user into the system's model of the current state of the task.

4.4.3. Communication with a Planning System. Problem solving can also be achieved through the use of a planning system that supports reasoning about goals, plans, and actions. While the task structures used in the Circuit-Fix-It Shop system involve task decomposition into subtasks and subsequently into primitive actions to be carried out, the problem-solving mechanisms are different from those that are used in conventional planning systems. The Circuit-Fix-It Shop system is presented with an explicit goal at the beginning of the dialogue and its task is to collaborate with the user in proving the goal in much the same way as a theorem is proved. Planning systems incorporate further complications in that often the system has to infer the user's goals from statements or actions that may not explicitly represent the goals (*plan recognition*). Planning systems typically include an explicit representation of beliefs, desires, and intentions that are reasoned about during the course of the problem solving. These elements are assumed implicitly in the Circuit-Fix-It Shop system.

The TRAINS project [Allen et al. 1995] is concerned with the integration of natural language dialogue and plan reasoning to support collaborative problem solving. The purpose of the dialogue is to negotiate and develop a plan. The speech acts that comprise the dialogue are motivated by reasoning about the plan and are at the same time interpreted in the light of the current plan. Figure 10 provides a simplified view of the components of the TRAINS system.

Plan reasoning in the TRAINS system involves two algorithms—the *incorporation* algorithm and the *elaboration* algorithm. The *incorporation* algorithm is concerned essentially with plan recognition,

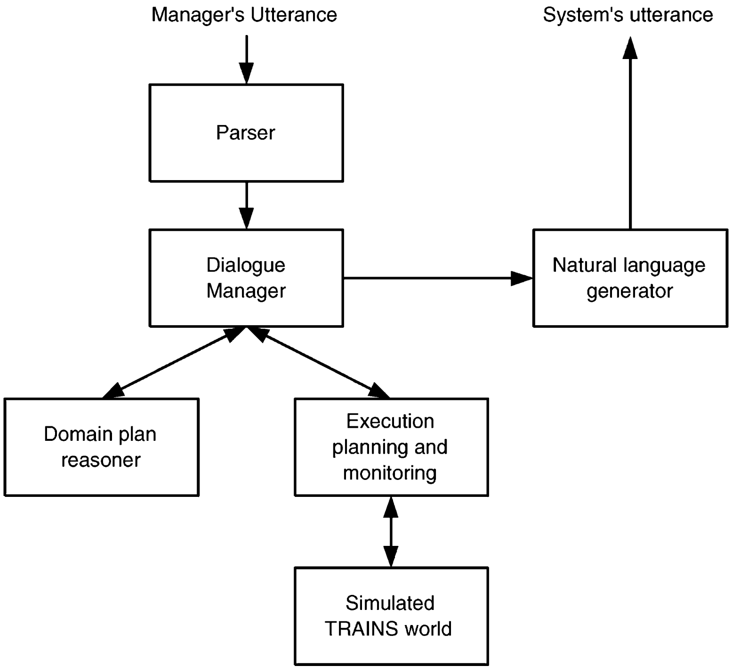


Fig. 10. The TRAINS architecture.

that is, with finding causal and motivational connections between potential interpretations of the current utterance and the current plan. The algorithm searches through a space of plan graphs with nodes representing events and states, and links representing relations between events and states such as enablement, effect, generation, and justification. The *elaboration* algorithm supports the system's construction of a plan using means-ends planning. If the user encounters some choice that requires confirmation, for example, an element in the plan that is ambiguous, the system generates an utterance to request confirmation.

4.4.4. Summary. This discussion of the role of the external communication component in a spoken dialogue system has shown how an integrated system architecture, as illustrated in Figure 7, is required in order to support interaction between the dialogue management component and the other system components. In addition to the problem of determining whether sufficient information has been elicited from the user to provide input to the external application, as discussed in Section 4.3, obtaining the required information from the external source is not necessarily a straightforward task and complex interactions may be required involving mediations between the dialogue manager and the user. In the case of a database query the requested information may not be available in the form that was requested so that a reformulated query is required. In a plan reasoning application such as TRAINS the plan reasoner may fail to find a connection between an event, goal, or fact inferred from the user's utterance and a node in the plan graph, in which case it could be assumed that the user's utterance had been misinterpreted and the language understanding component would be required to search for an alternative interpretation, failing which the system would request clarification or repair. Thus the interpretation and resolution of the user's query may involve complex interaction with the external source before the system can report a result back to the user.

4.5. Response Generation

Assuming that the requested information has been retrieved from the external source, the response generation component now has to construct the message that is to be sent to the speech output component to be spoken to the user. Broadly speaking, the construction of the message consists of three decisions involving:

(1) what information should be included;
(2) how the information should be structured;
(3) the form of the message—for example, the choice of words and syntactic structure.

Response generation can be achieved using simple methods, such as the insertion of the retrieved data into predefined slots in a template. On the other hand, complex methods using natural language generation techniques may be used, although generally these more complex methods have only been applied in research prototype systems.

Response generation in a dialogue system involves additional tasks beyond those required for other language generation tasks. Given that the information to be generated is in the form of some nonlinguistic representation—for example, the results of a database query or a chain of reasoning from an expert system—the dialogue manager has to relate the information to what was previously said (using a discourse history) as well as to the user's goals and knowledge (using a user model).

Use of a discourse history enables the system to provide a response that is consistent and coherent with the preceding dialogue. For example: if some entity that has already been mentioned is to be referred to again, the system should check whether an anaphoric expression can be used unambiguously to refer to the entity on a second mention, as in the following example taken from Reiter and Dale [1997]:

> *The next train is the* **Caledonian Express. It** *leaves at 10 am. Many tourist guidebooks highly recommend* **this train**.

Little research has been done on the use of pronouns in language generation, although there has been some research on generating definite descriptions—for example, the use of *the train* if the Caledonian Express and no other train has been previously mentioned [Dale and Reiter 1995].

As mentioned earlier, user modeling in the early 1980s was concerned with making natural language dialogue systems more cooperative. In addition to supporting the interpretation of the user's utterances by modeling the user's beliefs, goals, and plans, the other main application of user models was to enable a system to adapt its output to the user's perceived needs [Wahlster and Kobsa 1989]. A number of research projects addressed this issue, of which the following are indicative.

The KNOME system [Chin 1989] provided different levels of explanation of Unix commands depending on its categorization of the user's level of competence and the degree of difficulty of the command in question. The TAILOR system [Paris 1989] adapted its output to the user's level of expertise by selecting the type of description and the particular information that would be appropriate for a given user. Based on an extensive analysis of scientific texts, it was found that texts from adult encyclopedias and manuals for experts mainly included structural information that could be represented using constituency schemas describing the parts of the objects, while encyclopedias for young children and manuals for novices contained mainly process-oriented information that described the functional characteristics of the objects. TAILOR was able to generate appropriate descriptions to different types of users and to produce a range of descriptions for users falling between the two extremes of novice or expert. Finally, in the IMP system, Jameson [1989] investigated the use of anticipation feedback to determine the bias of the system's output. Basically what this involves is that the system attempts to anticipate the user's reaction to its output and then takes this anticipated reaction into account in finalizing its output. This technique is particularly appropriate for evaluation-oriented dialogues, such as personnel selection interviews and dialogues involving travel agents, hotel managers, and sales people.

A user model was used in the Circuit-Fix-It Shop system to enable the system to determine what needed to be said to the user and what could be omitted because of existing user knowledge (see the example discussed in Section 3.3). In this system the dialogue controller invoked inferences to derive additional axioms about the user based on the user's utterances. These inferences, which are similar to those used by Chin [1989] in the KNOME system, included the following [Smith and Hipp 1994], p. 60):

> *If the axiom meaning is that the user has a goal to learn some information, then conclude that the user does not know about the information. If the axiom meaning is that an action was completed, then conclude that the user knows how to perform the action.*

These inferences, which are based on abstract descriptions of actions and their effects, were used to provide user model axioms that could be used by the theorem prover along with other axioms that were available to prove goal completion. Thus the user model information was employed within the dialogue system to determine the selection of the information to be presented to the user.

A considerable amount of research in text generation has been concerned with the organization of messages, that is, their discourse structure. One of the most widely known approaches involves the use of rhetorical relations between elements of a text, as described in Rhetorical Structure Theory (RST) [Mann and Thompson 1988]. Examples of rhetorical relations are *elaboration*, *exemplification*, and *contrast*. Alternatively, schemas have been used to provide the structure of the information to be presented [McKeown 1985]. A schema sets out the main components of a text, using elements such as *identification*, *analogy*, *comparison*, and *particular-illustration*, which have a sequential ordering in a text and can occur recursively. Schema-based systems often use general programming constructs such as local variables and conditional tests.

The form of the output is known as the *linguistic realization*. This involves the choices of lexical items and syntactic structures to express the desired meaning. The choice of lexical items might involve deciding between the words *leave* and *depart* to express the concept of "departure," while syntactic decisions might involve the choice of an active or a passive sentence [Reiter and Dale 1997]. Linguistic realization also involves the generation of grammatically correct structures, for example, selecting the appropriate tense and rules of agreement. From the perspective of the construction of a text, four different categories of content may be involved [Reiter and Dale 1997]:

(1) unchanging text, that is, parts of the message that are always present in the output text;
(2) directly-available data, that is, information that has been retrieved from a database or knowledge base;
(3) computable data, that is, information that is derived from the data as a result of some computation or reasoning (for example, the number of records found in the database for trains between two cities);
(4) unavailable data, that is, information that is not present in the data but which supplements the information (this is common in texts authored by humans, for example, extra information that a railway line may be blocked by snow).

A dialogue system may make use of at least the first three types, using unchanging text for the constant parts of a message, retrieved data to convey the information that was requested, and computable data to summarize the information or to require a more specific choice from the user.

4.6. Speech Output

Speech output involves the translation of the message constructed by the response generation component into spoken form. In the simplest cases prerecorded canned speech may be used, sometimes with spaces to be filled by retrieved or previously recorded samples, as in

> *You have a call from <Jason Smith>. Do you wish to take the call?*

in which most of the message is prerecorded and the element in angular brackets is either synthesized or played from a recorded sample. This method works well when the messages to be output are constant, but synthetic speech is required when the text is variable and unpredictable, when large amounts of information have to be processed and selections spoken out, and when consistency of voice is required. In these cases text to speech synthesis (TTS) is used.

Text-to-speech synthesis can be seen as a two stage process involving

(1) text analysis;
(2) speech generation [Edgington et al. 1996a, 1996b].

Text analysis involves the analysis of the input text that results in a linguistic representation that can be used by the speech generation stage to produce synthetic speech by synthesizing a speech waveform from the linguistic representation. The text analysis stage is sometimes referred to as *text-to-phoneme conversion*, although this description does not cover the analysis of linguistic structure that is involved. The second stage, which is often referred to as *phoneme-to-speech conversion*, involves the generation of a prosodic description (including rhythm and intonation), followed by speech generation that produces the final speech waveform. A considerable amount of research has been carried out in text-to-speech synthesis which is beyond the scope of the present survey (see, for example, Edgington et al. [1996a, 1996b] and Carlson and Granström [1997] for recent overviews). This research has resulted in several commercially available text-to-speech systems, such as DECTalk and the BT Laureate system [Page and Breen 1996]. The main aspects of text-to-speech synthesis that are relevant to spoken dialogue systems will be reviewed briefly. The text analysis stage of text-to-speech synthesis comprises four tasks:

(1) text segmentation and normalization;
(2) morphological analysis;
(3) syntactic tagging and parsing;
(4) the modeling of continuous speech effects.

Text segmentation is concerned with the separation of the text into units such as paragraphs and sentences. In some cases this structure will already exist in the retrieved text, but there are many instances of ambiguous markers. For example, a full stop may be taken as a marker of a sentence boundary, but it is also used for several other functions such as marking an abbreviation (*St.*), as a component of a date (*12.9.97*), or as part of an acronym (*M.I.5*). Normalization involves the interpretation of abbreviations and other standard forms such as dates, times, and currencies, and their conversion into a form that can be spoken. In many cases ambiguity in the expressions has to be resolved—for example, *St.* can be "street" or "saint."

Morphological analysis is required, on the one hand, to deal with the problem of storing pronunciations of large numbers of words that are morphological variants of one another, and, on the other, to assist with pronunciation. Typically a pronunciation dictionary will store only the root forms of words, such as *write*. The pronunciations of related forms, such as *writes* and *writing*, can be derived using morphological rules. Similarly, words such as *staring* need to be analyzed morphologically to establish their pronunciation. Potential root forms are $star + ing$ and $stare + ing$. The former is incorrect on the basis of a morphological rule that requires consonant doubling (*starring*), while the latter is correct because of the rule that requires *e*-deletion before the *-ing* form.

Tagging is required to determine the parts of speech of the words in the text and to permit a limited syntactic analysis, usually involving stochastic processing. A small number of words—estimated at between 1 and 2% of words in a typical lexicon [Edgington et al. 1996a]—have alternative pronunciations depending on their part of speech. For example: *live* as a verb will rhyme with *give*, but as an adjective rhymes with *five*. The part of speech also affects stress assignment within a word—for example, *record* as a noun is pronounced *'record* (with the stress on the first syllable), and as a verb as *re'cord* (with the stress on the second syllable).

Modeling continuous speech effects is concerned with achieving natural sounding speech when the words are spoken in a continuous sequence. Two problems are encountered. First, there are weak forms of words, involving mainly function words such as auxiliary verbs, determiners, and prepositions. These words are often unstressed and given reduced or amended articulations in continuous speech. Without these adjustments the output sounds stilted and unnatural. The second problem involves coarticulation effects across word boundaries, which have the effect of deleting or changing sounds. For example: if the words *good* and *boy* are spoken together quickly, the /d/ in *good* is assimilated to the /b/ in *boy*. Modeling these coarticulation effects is important for the production of naturally sounding speech.

There has been an increasing concern with the generation of prosody in speech synthesis, as poor prosody is often seen as a major problem for speech systems that tend to sound unnatural despite good modeling of the individual units of sound. Prosody includes phrasing, pitch, loudness, tempo, and rhythm, and is used to convey differences in meaning as well as to convey attitude.

The speech generation process involves mapping from an abstract linguistic representation of the text, as provided by the text analysis stage, to a parametric continuous representation. Two main methods have been used to model speech: *articulatory synthesis*, which models characteristics of the vocal tract and speech articulators, and *formant synthesis*, which models characteristics of the acoustic signal. Formant synthesis has been the more successful method and has produced commercial systems such as DECTalk that yield a high degree of intelligibility.

An alternative method that is used in recent work, for example, in BT's Laureate system, involves concatenative speech synthesis, in which prerecorded units of speech are stored in a speech database and selected and joined together in speech generation. The relevant units are usually not phonemes, due to the problems that arise with coarticulation, but diphones, which assist in the modeling of the transitions from one unit of sound to the next. Various algorithms have been developed for joining the units together smoothly.

Generally relatively little emphasis has been put on the speech output process by developers of spoken dialogue systems. This is partly due to the fact that text-to-speech systems are commercially available that can be used to produce reasonably intelligible output. However, there are certain applications where more naturally sounding output is desirable, for example, in applications involving the synthesis of speech for the handicapped, or in systems for foreign language instruction.

4.7. Summary

The basic components of spoken dialogue systems that have been described in the preceding sections represent various technologies, each of which constitutes a major research and development area in its own right. An interesting aspect of spoken dialogue systems is that these separate technologies have to be somehow harnessed and integrated to produce an acceptable, working system. It is essential that the components of the system should work in integration—indeed, the efficiency of the individual components is less important than the efficiency of the complete system. For example, it can be argued that a system with a high-performance speech recognizer would still be ineffective if the dialogue control component functioned poorly. Conversely, a good dialogue control component can often help compensate for the weaknesses of the speech recognizer by producing a reasonable response in the face of unreliable input. One of the major challenges for developers of spoken dialogue systems is to integrate the component technologies to produce a robust and acceptable system, in which the whole is greater than the sum of its parts.

5. DIALOGUE CONTROL

There are two aspects to dialogue control: the extent to which one of the agents maintains the initiative in the dialogue and the ways in which the flow of the dialogue is managed. Dialogue control may be system-led, user-led, or mixed-initiative. In a system-led dialogue the system asks a sequence of questions to elicit the required parameters of the task from the user. In a user-led dialogue the user controls the dialogue and asks the system questions in order to obtain information. In a mixed-initiative dialogue control is shared. The user can ask questions at any time, but the system can also take control to elicit required information or to clarify unclear information. The management of dialogue control is not an issue for user-led dialogue as the user decides which questions to ask, as in Question-Answer and Natural Language Database Systems. In system-led and mixed-initiative dialogue the control has to be managed in order to determine what questions the system should ask, in what order, and when. Three main strategies for dialogue control were identified in Section 2.5 and illustrated in Section 3. Finite state-based dialogue control supports a system-led dialogue in which all the questions that the system asks have been predetermined. Frame-based systems are also primarily system-led, although they permit a limited degree of user initiative. Agent-based systems tend to be mixed-initiative. These distinctions along with other aspects of dialogue control will be examined in the following subsections.

5.1. Finite State-Based Systems

5.1.1. The Basic Model. In a basic finite state-based system the dialogue structure is represented in the form of a state transition network in which the nodes represent the system's questions and the transitions between the nodes determine all the possible paths through the network, thus specifying all legal dialogues. Each state represents an information state in which some information is elicited from or confirmed with the user. Subdialogues can be used within the basic network to support a more modular design approach and to provide libraries of commonly occurring transactions. Figure 11, taken from the Danish Dialogue Project, illustrates the use of a basic finite state network to model the dialogue flow for an automatic book club service [Larsen and Baeekgaard 1994]. In this dialogue the system progresses through a series of states, with the transitions between states being determined by the user's responses. There are various choice points and loops, as well as subdialogues (for example, for the tasks CHECK MEMBER, ORDER, CANCEL, and OVERVIEW). Furthermore, in this particular architecture, the user can use the key words *repeat* and *change*, to request repetition of the system's output and to change a previously accepted parameter.

5.1.2. Advantages of Finite State Models. A major advantage of the finite state model is its simplicity. From a developer's perspective state transition networks are particularly suitable for modeling dialogue flow in a well-structured task involving information to be exchanged in a predetermined sequence, with the system retaining control over the dialogue and deciding which question to ask next. In this way the semantics of the system is clear and intuitive. Moreover, as the user's responses are restricted, fewer technological demands are put on the system components, particularly the speech recognizer. The lack of flexibility and naturalness may be justified as a trade-off against these technological demands. For these reasons

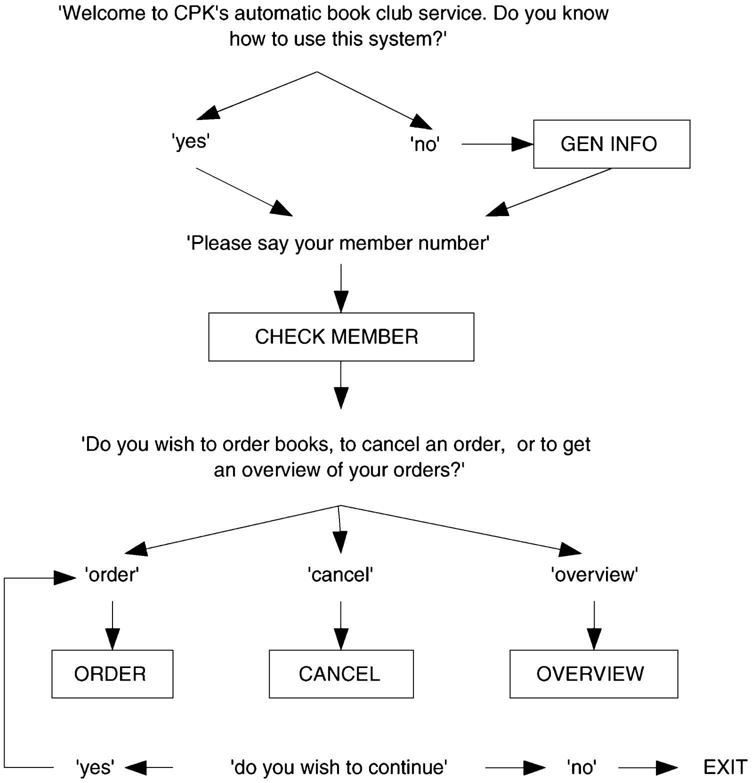

'Welcome to CPK's automatic book club service. Do you know how to use this system?'
'yes'
'no'
GEN INFO
'Please say your member number'
CHECK MEMBER
'Do you wish to order books, to cancel an order, or to get an overview of your orders?'
'order'
'cancel'
'overview'
ORDER
CANCEL
OVERVIEW
'yes'
'do you wish to continue'
'no'
EXIT

Fig. 11. Dialogue graph for an automatic book service.

most currently available commercial systems use some form of finite state dialogue modeling.

It is interesting to note that there is some support in empirical studies for the use of state-based dialogue control. Hone and Baber [1995] examined the relationship between dialogue control and transaction times, finding that more constrained dialogues that employed a menu-like interaction style with *yes*/*no* confirmation of all user input tended to result in dialogues with longer transaction times, as would be expected. However, this effect depended on the system's level of recognition accuracy, which was manipulated in the experiments. It was found that there was a greater likelihood of errors in the less constrained system as it permitted a larger active recognition vocabulary.

In another study two versions of a simple call assistance application were built [Potjer et al. 1996]. The system-led version used isolated word recognition and word spotting, while the mixed-initiative version used continuous speech recognition and more complex natural language processing. In the system-led version the user was prompted for the required service in two steps, while in the mixed-initiative version the user could request the service in a single utterance. The minimum number of turns per transaction was lower for the mixed-initiative system, although more additional turns were required for the mixed-initiative system on account of the greater number of recognition errors. Thus the system-led interface was not slower than its mixed-initiative counterpart. Moreover, a subjective analysis of user satisfaction indicated that users were satisfied with both versions.

Similar results were found in a study involving train timetable information in which it was found that for simple services a system-driven dialogue using isolated word recognition achieved good user acceptance [Billi et al. 1996]. This finding was supported in a study of dialogue strategies comparing explicit and implicit recovery from communication breakdowns [Danieli and Gerbino 1995]. The version incorporating explicit confirmation and repair, which made greater use of isolated word recognition and spelling, was found to be robust and safe, even though it increased the number of turns required to complete the transaction. The conclusion from these studies is that system-led dialogue using state transitions would appear to be suitable for simple tasks with a flat menu structure and a small list of options, bringing also the advantage of less complex spoken language and dialogue modeling technology. The lack of flexibility and naturalness may be justified as a trade-off against these technological demands.

As mentioned earlier, state transition networks are particularly suitable for modeling dialogue flow in well-structured tasks. The automatic book service illustrated in Figure 11 is a good example. Other examples are directory assistance, questionnaires, and travel inquiries, provided the dialogue is constrained to a basic, system-led series of questions to elicit a number of well-defined responses. Consider the part of a directory inquiry dialogue in which the system elicits the name of the person to be called: here the system has to identify a unique individual, which generally requires eliciting a first and last name. This might be accomplished in a single step—*Request First and Last Name*—or in a series of steps—*Request Surname* > *Request Spelling of Surname* > *Request First Name* > *Confirm First and Last Name*. A finite state dialogue model could be created for this task with subdialogues for subtasks such as requesting the surname and first name. Additional states would be required for cases of multiple individuals with the same name, variations on first names, and names that are pronounced similarly (homophones) and thus require spelling to disambiguate. The main characteristic of this task is that there is a finite and clearly defined set of information items to be exchanged, the information can be elicited in a natural order, and the task may be decomposed into a hierarchy of well-ordered subtasks [McTear 1998]. A finite state model could also be used for similarly structured tasks such as obtaining weather forecasts or football scores, ordering items from a catalogue, or making simple bank transactions.

Dialogues for questionnaires are also highly structured even though a large number of questions may be required to elicit the required information. For questionnaires the user can be constrained through carefully designed prompts to produce an acceptable range of responses [Hansen et al. 1996]. In a large-scale project involving the U.S. Census the dialogue was implemented using a finite state network as the information had to be elicited in a fixed order, for example: *Name* > *Gender* > *Birth date* > *Marital Status*, etc., with subdialogues being used for the more complex items ([Cole et al. 1997]. Finite state models can be used for similar tasks such as eliciting a person's personal details for financial transactions or obtaining information for insurance quotes. The key characteristic of this class of dialogues is that they are well structured. Even though there may be several items of information to be elicited, these can be broken down into well-structured subtasks that are independent of one another.

5.1.3. Disadvantages of Finite State Networks. Finite state dialogue models are not suitable for modeling less well-structured tasks characterized by subtasks whose order is difficult to predict, by information modeled at different levels of abstraction, or by complex dependencies between items of information [Kamm 1995]. A good example is the Flight Reservation System of the Danish Dialogue Project [Dybkjær et al. 1998]. Although the reservation task would appear to be well structured, as it consists of a series of ordered subtasks, there are complex dependencies in this system between various parameters, for example, between discounted fares and flight availability. As a result a client could opt for a discounted fare and go on to confirm several parameters only to have to backtrack to a different dialogue path because the desired departure time was not available at the discounted price. The key word *change* can be spoken by the user of this system to correct the latest piece of information given to the system, but to correct earlier information *change* has to be used repeatedly to cause the system to backtrack sequentially until the item to be changed is reached. Thus, when there are dependencies between the items of information, the use of a finite state dialogue model becomes unwieldy, leading to a combinatorial explosion of states and transitions.

Finite state dialogue models are inflexible. This characteristic is not a problem if the interaction with the user is controlled by the system and restricted to a well-ordered sequence of questions. However, because the dialogue paths are specified in advance, there is no way of managing deviations from these paths. Problems arise if the user needs to correct an item or introduce some information that was not foreseen at the time of the design of the dialogue. Adding natural language facilities, while providing the user with greater flexibility in what they can say, can add to these problems. Taking the example of a simple travel inquiry system, a natural order for the system's questions might be: *destination* > *origin* > *date* > *time*. However, when answering the system's question concerning destination the user might reply with a destination as well as the departure time (or indeed other combinations of the four required parameters). A finite state-based system would simply progress through its set of predetermined questions, ignoring or failing to process the additional information and then asking an irrelevant question concerning the departure time.

The solution to this problem would be to include a dialogue model so that the system "knows" what it has already elicited as well as what has still to be asked. The system could then loop through the dialogue model until all the required information has been elicited. In this way the problem of irrelevant questions would also be avoided. However, the problem is that, as soon as the number of items grows, the number of transitions to cater for each required dialogue path grows to unmanageable proportions. This problem is further augmented if adequate repair mechanisms are to be included at each node for confirmation or clarification of the user's input. Thus it was estimated that in the Philips system there were about 1,000 system questions. Allowing for flexible adaptation to the user's input—for example, in the case where a user says more than the system expected or provides an unanticipated response—given that almost any question could follow almost any other—the network would require tens of thousands of transitions [Aust and Oerder 1995].

Dialogues involving some form of negotiation between system and user cannot be modeled using finite state methods, as the course of the dialogue cannot be determined in advance. For example, planning a journey may require the discussion of constraints that are unknown by either the system or the user at the outset. In these interactions some form of negotiation and discussion of constraints is required. For example, in the TRAINS project, to be discussed below, the user and the system collaborate to construct an agreed upon executable plan that has to be developed incrementally in order to incorporate new constraints that arise during the course of the dialogue [Allen et al. 1995].

5.2. Frame-Based Systems

Rather than build a dialogue according to a predetermined sequence of questions to be asked, a frame-based system takes the analogy of a form-filling task in which a predetermined set of information is to be gathered. This frame (or template) fulfils the role of a dialogue model that keeps account of the items for which the system requires information. Naturally this will also involve questions, but the questions do not have to be asked in a particular sequence. For example, in the Philips train timetable system, the questions that the system might ask are listed together with their preconditions—that is, the conditions under which that question should be asked. Some questions for a travel system might be

condition: unknown(origin) & unknown (destination)
question: "Which route do you want to travel?"
condition: unknown(origin)
question: "Where do you want to travel from?"
condition: unknown(destination)
question: "Where do you want to travel to?"

Given all the questions and their preconditions, which do not need to be stated in chronological order, the dialogue control component can decide the next question to be asked based on those questions whose preconditions are true. If several questions can be asked at a particular stage in the dialogue, other factors can be used to choose a question to be asked. For example, in the Philips SpeechMania system, each dialogue action (including the questions) is coded with a key word that determines the dialogue action's priority and thus the dialogue flow. Some examples of these key words in their default order of priority are

ONCE:	for an action that has not yet been executed, for example, the initial greeting;
MULTIPLE:	if more than one value has been returned for a variable, so that ambiguity resolution is required;
VERIFIABLE:	to be used if a value has not been confirmed by the user;
UNDEFINED:	to be used when no value has been defined for a variable and a question is required to elicit the value from the user.

Given this priority mechanism, problems relating to what is ambiguous are resolved before attempts to verify a value, which are in turn resolved before questions for new values. Thus a sequence of questions evolves based on the current context of the system (what has been asked so far, what information is ambiguous, what has to be confirmed), without having to specify predetermined paths through a dialogue network.

A similar mechanism has been used in the Communicator system developed at the University of Colorado, Boulder [Ward and Pellom 1999]. This system obtains information from the Internet on airline flights, hotels, and rental cars. The dialogue control is described as "event-driven," meaning that the dialogue manager decides what to do next based on the current system context rather than a predetermined script. In this case the context consists of the semantic content of the user's input together with a template of slots to be filled. On assimilating a parsed user's utterance with the dialogue context, the system decides on its next action according to a set of priorities similar to those used in the Philips system:

—clarify if necessary;
—finish if all done;
—retrieve data and present to user;
—prompt user for required information.

A variation on frames is the use of a form consisting of a number of slots for the relevant attributes in the domain. Dahlbäck and Jönsson [1999] described their use of information specification forms for a bus timetable information system. Forms are also used as the main dialogue items in VoiceXML documents. A form in VoiceXML consists of field and control items. A field gathers information from the user using speech or DTMF input while control items involve sequences of procedural statements for prompting and computation. The Form Interpretation Algorithm determines which items in a form to visit depending on the status of their guard condition. Thus, unless a field variable within a form has the value *undefined*, that form will not be visited. In a directed form the form items are executed once in a sequential order, resulting in a rigid, system-directed dialogue. A mixed-initiative form, combined with a grammar, enables the user to input all the required items in one utterance, giving a more flexible dialogue.

Goddeau et al. [1996] discussed a more complex type of form, the E-form (electronic form), which has been used in a spoken language interface to a database of classified advertisements for used cars. E-forms differ from the types of form and frame described so far, in that the slots may have different priorities for different users—for example, for some users the color of a car may be more critical than the model or mileage. Furthermore, information in slots can be related—for example, a more recent model usually costs more. The E-form allows users to explore multiple combinations to find the car that best suits their preferences. Thus the selection of an appropriate car is viewed as an optimization task which involves more than the retrieval of a set of records from a database. However, it is the user who performs this optimization, whereas in a problem-solving system the optimization would be performed by the system or, ideally, as a result of a negotiation dialogue between system and user. The E-form is used to determine the system's next response, which is based on the current status of the E-form, the most recent system prompt, and the number of items returned from the database:

- —If no records found, ask user to be more general.
- —If fewer than five records found, consider the search complete and generate a response that outputs the retrieved records.
- —Otherwise cycle through an ordered list of prompts choosing the first prompt whose slot in the E-form is empty.
- —If too many records have been found and all the prompt fields have been filled, ask the user to be more specific.

Other data structures that can be used to control the dialogue are schemas, task structure graphs, and type hierarchies. Schemas are used in the Carnegie Mellon Communicator system to model more complex tasks than the basic information retrieval tasks that use forms [Constantinides et al. 1998; Rudnicky et al. 1999]. A schema is a strategy for completing a goal in a task-based dialogue, such as determining an itinerary. The itinerary is represented as a hierarchical data structure that is constructed interactively over the course of the dialogue. At the same time the nodes in the tree are filled with specific information about the trip. While there is a default sequence of actions to populate the tree that is maintained as a stack-based agenda, the user and the system can both control this ordering and cause the focus of the dialogue to shift (for example: *Let's talk about the first leg [of the itinerary] again*). Task structure graphs provide a similar semantic structure to the E-form and are used to determine the behavior of the dialogue control module as well as the language understanding module [Wright et al. 1998]. The graph depicts relationships between the elements of a customer-services application and is used to provide a contextual interpretation of spoken utterances in a dialogue. Similarly, type hierarchies can be used to model the domain of a dialogue and as a basis for clarification questions [Denecke and Waibel 1997]. Given that information in a type hierarchy can be missing or underspecified, clarification requests are generated to enable the user to achieve their communicative goal.

In summary: there are a number of different types of data structure, such as the frame, E-form, schema, task structure graph, and type hierarchy that can be used to model the structure of the information required by the user and to determine the actions to be taken by the dialogue system to obtain this information.

5.2.1. Advantages of Frame-Based Systems.

The frame-based approach has several advantages over the finite-state-based approach, for the user as well as the developer. As far as the user is concerned, there is greater flexibility. For example, there is some evidence that it can be difficult to constrain users to the responses required by the system, even when the system prompts have been carefully designed to do just that [Eckert et al. 1995]. The ability to use natural language and the use of multiple slot filling enables the system to process the user's overinformative answers and corrections. In this way the transaction time for the dialogue can be reduced, resulting in a more efficient and more natural dialogue flow. The frame-based system fulfilled a number of dialogue design requirements identified by the Philips dialogue team, including the following:

—there should not be a rigid question-answer scheme to obtain the required values;

—no more questions than necessary should be asked;

—no more confirmation than necessary should be required;

—information given by the caller, prior to the system asking for it, should be used [Philips Speech Processing 1997].

Similarly, the Communicator system developed at the University of Colorado [Ward and Pellom 1999] enables a mixed-initiative dialogue in which the user can take control. This degree of user control is greater than in the Philips system, where the system has control of the dialogue flow but the user can insert corrections to items that the system has misrecognized or misunderstood. In the Communicator system the user can respond with anything to the system's question, that is, not necessarily the answer to the question. The system will parse the utterance and decide whether and how to respond to it, putting on hold prompts for any additional missing information that is required by the system to build a database query.

From a developer's perspective, implementing this degree of flexibility in a graph-based system becomes cumbersome, if not impossible. A large number of states and transitions are required to deal with the number of different paths that dialogues might take. A frame-based system can be specified declaratively with the system's questions listed as in a rule-based expert system (or production system).

5.2.2. Disadvantages of Frame-Based Systems.

Finite state-based and frame-based approaches are appropriate for well-defined tasks in which the system takes the initiative in the dialogue and elicits information from the user to complete a task, such as performing a database query. Frame-based systems provide greater flexibility than state-based systems as the dialogue flow is event-driven and not predetermined. However, the system context that contributes to the determination of the system's next action is fairly limited, being confined essentially to the analysis of the user's previous utterance in conjunction with a template of slots to be filled and a number of priorities for control of the dialogue. More complex transactions cannot be modeled using these approaches for the following reasons:

—different users may vary in the level of knowledge they bring to the task, so that a wide range of responses is required by the system;

—the state of the world may change dynamically during the course of the dialogue, with the result that it is not possible to specify all possible configurations in advance;

—the aim of the dialogue is not just to obtain sufficient information from the user to execute a database query or carry out some action—instead, the dialogue involves the negotiation of some task to be achieved, involving planning and other types of collaborative interaction.

For the developer a frame-based approach has the disadvantage of any production system with a large number of rules and contexts in that it is difficult to predict which rule (or question) is likely to fire in a particular context. A considerable amount of experimentation may be required to ensure that the system does not produce an inappropriate question under some circumstances that had not been foreseen at design time.

5.3. Agent-Based Systems

Agent-based approaches draw on techniques from Artificial Intelligence (AI) and focus on the modeling of dialogue as collaboration between intelligent agents.

Several classes of agent-based system will be described and evaluated, including systems using theorem proving, planning, distributed architectures, and conversational agents.

5.3.1. Dialogue Control Using Theorem Proving. The functionality of the Circuit-Fix-It Shop system was illustrated in Section 3.3 and its architecture presented in Section 4.4.2. A key aspect of the system is that solutions are developed dynamically, on the basis of task steps recommended by the domain processor, taking into account the current situation and the user's current state of knowledge. This dynamic development contrasts with the approach adopted in some plan-based systems, in which a complete solution is developed by the system and then communicated to the user. Theorem proving is used in the Circuit-Fix-It Shop system to determine task completion, and dialogue is required for the acquisition of axioms that are missing but are required to complete a task step. Thus dialogue is integrated closely with task processing and is invoked when the system is unable to complete the task using its own resources. A brief overview of the theorem-proving approach and of the role of missing axioms in the production of dialogue is presented in the following paragraphs.

The role of the theorem prover is to determine task completion. The dialogue controller receives a suggested task (or goal) from the domain processor and selects which goal is to be solved. The dialogue controller decides when theorem proving is to be activated, whether language can be used to acquire missing axioms, and when theorems must be dynamically modified. The following example (discussed earlier in Section 3.3) illustrates this process.

The task to be completed involves the system finding out whether there is a wire between connectors 84 and 99, represented in GADL (Goal and Action Description Language) as

> *goal(computer,learn(ext_know(phys_state (prop(wire(84,99),exist,X),true)))).*

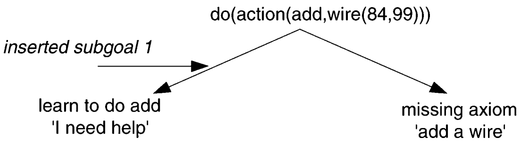


Fig. 12. Inserting a substep into the proof tree.

X represents an unknown value for the existence of the wire. One way to solve this goal is to look for an appropriate axiom in the knowledge base, such as:

> *axiom(phys_state(prop(wire(84,99),exist, present),true))*—that is, wire is present,

or

> *axiom(phys_state(prop(wire(84,99),exist, absent),true))*—that is, wire is absent.

As there is no appropriate axiom in the knowledge base, the system determines that the next step should be for the user to add the wire, which should then provide the missing axiom that the wire is present. This step is conveyed by the system to the user using language. However, at this point the user is unable to complete this task step and requests help. This response does not satisfy the missing axiom. In conventional theorem proving, a new rule for the proof would be sought using backtracking. However, for this to be successful, theorem descriptions would have to be listed for every possible response that provides an axiom other than the required one. A more general approach is to modify the original theorem by inserting the substep that needs to be resolved (in this case, how to add the wire), and then resume the theorem-proving process once the substep has been solved. To do this, an Interruptible Prolog SIMulator (IPSIM) is employed to enable the dynamic modification of theorems. IPSIM inserts into the active theorem specification the required substep (learning how to add the wire) before the theorem step of acquiring the required axiom about adding the wire (see Figure 12).

As can be seen, the substep of learning how to add the wire has to be solved before the missing axiom of adding the wire can be acquired. Thus the problem solving that is involved in the Circuit-Fix-it Shop system is accomplished using theorem proving, while the Missing Axiom theory provides the mechanism for using language. The interruptible theorem prover creates subdialogues to solve substeps that are inserted into the partially completed proof. Given the overall control exerted by the dialogue controller, it is possible to select which substep to work on first and also to move between substeps (or subdialogues) if this is appropriate given the current state of the proof, the system's inferences concerning the user's knowledge, and the user's responses. In this way, dialogue is closely integrated with problem solving, but the theorem-proving paradigm that is used with the interruptible theorem prover allows the dialogue to evolve dynamically.

5.3.2. Plan-Based Approaches. In plan-based approaches to dialogue, utterances are treated in the same way as actions in a planning system that are performed in order to achieve some goal [Cohen 1994]. The goal of the utterance may be some desired physical state, such as having a drink of beer in a bar. In this case, an utterance that functions as a request for a beer is incorporated into a plan that also involves physical actions, such as the customer handing over some money and the bartender giving the beer. On the other hand, an utterance that has the function of conveying information effects a change in the listener's mental state. Much of the early work involving plans in the 1980s was concerned with recognizing the intention behind an utterance and matching this intention with some part of a plan that might achieve a particular goal [Allen 1983]. A cooperative system would adopt the user's goal, anticipate any obstacles to the plan, and produce a response that would promote the completion of the goal [Allen and Perrault 1980].

A key element of this approach was the modeling of utterances as speech acts, which, like action operators in planning, consisted of roles, preconditions, constraints, and effects. For example: the following is a definition of the communicative act *ConvinceByInform* [Allen 1995]:

Roles: Speaker, Hearer, Prop
Constraints: Agent(Speaker), Agent(Hearer), Proposition(Prop), Bel(Speaker,Prop)
Preconditions: At(Speaker, Loc(Hearer))
Effects: Bel(Hearer,Prop)

With this communicative act, if a speaker informs a hearer of some proposition and convinces the hearer of that proposition, one constraint (a condition that must be true) is that the speaker must believe the proposition. A precondition of the act—which if not true, can be made true through a further action—is that the speaker should be at the same location as the hearer (for face-to-face communication). As a result of the communicative act, the hearer will believe the proposition. A plan to achieve a goal involving language would typically involve chaining together a series of such communicative acts, including acts that involve representing the intentions and communicative actions of another agent. Allen [1995] provided a detailed account of the planning underlying a simple dialogue concerned with the purchase of a train ticket. Part of this plan involves the agent purchasing the ticket finding out the price of the ticket from the clerk, which in turn requires the clerk to produce a ConvinceByInform act stating the price of the ticket. But to achieve this, the agent buying the ticket has to produce a MotivateByRequest act that has as its effect an intention on the part of the clerk to produce the required ConvinceByInform act. These acts are chained together in the correct sequence to achieve the goal.

There are several problems with plan-based approaches. In order to infer a speaker's plan, the listening agent has to be able to recognize the communicative act performed by the speaker's utterance and locate this act within a particular plan schema. If the communicative act has been incorrectly recognized, this will result in an incorrect identification of the speaker's plan. At the very least this will require additional mechanisms for repairing the incorrect assignment of the speaker's intention. In addition to this, however, there is the problem that the processes of plan recognition and planning, which involve chaining from preconditions of plans to actions, can in more complex cases become combinatorially intractable. As planning algorithms require reasoning from first principles, they are best suited for restricted domains in which the reasoning is kept to manageable proportions.

Much of the early work in planning focused on the analysis of individual utterances rather than on how these utterances could be combined to form a coherent dialogue. Litman and Allen [1987] extended the basic model in two directions. First, they introduced the notion of a hierarchy of plans, which included subdialogues for clarification or correction of a plan. They also introduced a distinction between domain plans and discourse plans. Domain plans, which involve task-related speech acts, are modelled using traditional plan-based approaches. Discourse plans, which are task-independent, involve acts used to control the dialogue such as "clarification" and "correction." However, this approach still required an assumption of cooperativity between the agents and did not provide an explanation of the dialogue activity of agents who did not share mutual goals.

Some alternative approaches that address these issues will be presented in the next subsections. One direction, as illustrated in the TRAINS system, extends the traditional plan-based approach by modeling conversational agency within a multiagent action theory. A second direction, implemented in the SUNDIAL system, does not attempt to infer a speaker's intentions but bases decisions about dialogue continuation on the current state of the dialogue and the system's current belief state. Finally, a third approach, as exemplified by the ARTEMIS system, models dialogue as a process of rational interaction in which instances of dialogue structure emerge as a consequence of the dynamics of rationality principles.

5.3.3. Conversational Agency in the TRAINS System. The TRAINS project has extended early work in plan-based systems in two main directions. First, the approach to problem solving has required more elaborate reasoning to support collaborative and incremental planning. This is to enable the mixed-initiative planning that is usually involved when two agents collaborate to solve a problem. In the TRAINS project the system has the ability to plan low-level details such as routes between cities and has knowledge of constraints such as congestion at certain stations or whether an engine is available. The human manager, on the other hand, has knowledge of high-level goals, is aware of the motivations and justifications of particular plans of action, and can decide whether to relax soft constraints such as accepting a route that involves congestion in a particular city on the route. The system and the manager collaborate to construct an agreed upon executable plan. However, unlike in conventional planning where the initial goal is completely specified and a complete plan can be produced, in the TRAINS project the initial goal is typically underspecified and the plan has to be developed incrementally in order to incorporate new constraints that arise, such as information about congestion. As a result modifications to the original plan may be required. The general model for this mixed-initiative planning involves four steps [Ferguson et al. 1996]:

Focus: Identify the goal or subgoal under consideration.

Gather constraints: Collect constraints involving resources, background information, and preferences.

Instantiate solution: Generate a solution as soon as one can be produced efficiently.

Criticise, correct or accept: If the solution is criticized and modifications are required, continue with Step 2; otherwise, if the solution is acceptable, go to Step 1 and select a new goal or subgoal.

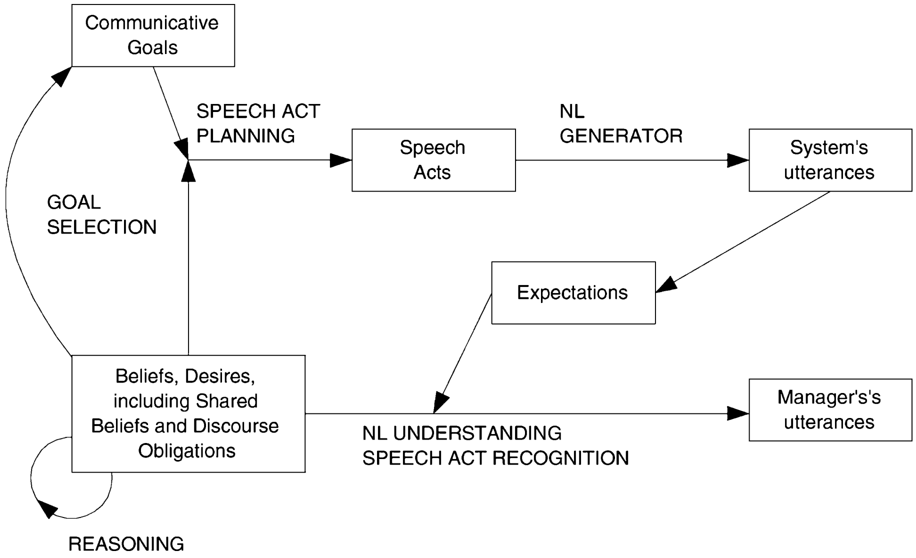


Fig. 13. The BDI model of conversational agency.

This model provides a basis for the tasks to be performed in the dialogue and the order in which they are to be performed. It would be possible to develop a corresponding task hierarchy and translate this directly into a dialogue structure. However, a second aim of the project was to develop a more elaborate model of dialogue that would provide the required flexibility for mixed-initiative interaction and would be suitably integrated with the complex commonsense knowledge and reasoning of the problem-solving component.

The dialogue manager in the TRAINS system functions as a conversational agent that performs communicative acts when it sends messages to other agents, observes the communicative acts of the other agents, and maintains and monitors its own mental state. Thus the model of dialogue in the TRAINS system is more ambitious than in most other systems where the dialogue component is a tool that provides a front-end interface to the other components of the system [Traum 1996]. The theoretical foundation of the conversational agent is provided by the BDI (Belief, Desire, and Intention) architecture of Bratman et al. [1988], which models agents that plan and execute actions in the physical world. The original BDI model is specialized to conversational actions, as shown in Figure 13 [Allen 1995]. Based on its current beliefs about the domain, including nested beliefs about shared knowledge, and the discourse obligations that each conversational agent has, the agent selects communicative goals, decides what speech act to perform next, generates an utterance, analyzes the manager's response, and updates its beliefs about the discourse state and its own discourse obligations.

Discourse obligations are an important element of the conversational agent. Earlier plan-based models of dialogue were based mainly on an analysis of the intentions of speakers. A dialogue agent would construct a model of a speaker's intentions, infer the speaker's goals, and adopt a goal to achieve these goals. However, this model made strong assumptions of cooperativeness and failed to explain why an agent would still respond when it did not know an answer or did not wish to adopt the speaker's goals. The solution was to make a distinction between the *intentions* that an agent might have in performing a task and the *obligations* that constitute the conventions of cooperative conversation [Traum and Allen 1994]. These discourse

Table II. Some Obligation Rules [Traum and Allen 1994]

Source of obligation	Obliged action
S_1 Accept or Promise A	S_1 achieve A
S_1 Request A	S_2 address Request: accept A or reject A
S_1 YesNo Question whether P	S_2 Answer-if P
S_1 WH-Question P(x)	S_2 Inform-ref x
utterance not understood or incorrect	repair utterance

obligations are socially based—they represent what an agent should normally do and they are generally addressed before task-related goals and intentions. Table II presents some discourse obligation rules. Thus, to take the example of a request by S_1, there is a discourse obligation on S_2 to respond to the request, though not necessarily to accept it.

In addition to these discourse obligations, the agent has to consider various general conversational obligations, such as acknowledging the utterances of others and not interrupting, coordinating mutual beliefs (grounding), and planning the execution of domain goals. The following discourse actor algorithm shows how these competing goals are prioritized [Traum and Allen 1994]:

```
while conversation is not finished
  if system has obligations
  then address obligations
  else if system has turn
  then if system has intended conversation acts
    then call generator to produce NL utterances
    else if some material is ungrounded
    then address grounding situation
    else if some proposal is not accepted
    then consider proposals
    else if high-level goals are unsatisfied
    then address goals
    else release turn or attempt to end conversation
  else if no one has turn
  then take turn
  else if long pause
  then take turn
```

This algorithm produces a reactive-deliberative model of dialogue agency in which the agent reasons about its discourse obligations and domain goals. Where there is a conflict, the agent displays a "relaxed" conversational style in which discourse obligations are addressed before the agent's own goals. This gives rise to an interaction in which the initiative of the other agent is followed. However, in a less cooperative situation, the agent can continue to address its discourse obligations but can respond in different ways, for example, by rejecting requests and refusing to answer questions. Finally, the system can take the initiative, for example, in a situation where the other agent does not take a turn, by taking the turn and using this opportunity to address its own goals. Thus, depending on the flow of the dialogue and on the behavior of the other agent, the TRAINS conversational agent can shift its focus from the obligation-driven process of following the other's initiative to the goal-driven process of taking the lead in the conversation.

5.3.4. Event-Driven Dialogue in a Distributed Architecture—the SUNDIAL Dialogue Manager. As mentioned earlier, plan-based approaches depend on the system being able to assign an intention to the user's utterances that can be related to the user's plan. An alternative approach models dialogue on the basis of a combination of the system's belief and intention states, and does not attempt to model these states in the user. This is the approach adopted in the SUNDIAL system [McGlashan et al. 1990], in which dialogue continuation is based on the results of a contextual semantic interpretation of the user's utterance and on monitoring changes in the system's belief state. This approach is event-driven in the same way as a frame-based system, but the difference is that the representation of context in SUNDIAL is more complex than that used in frame-based systems. SUNDIAL uses a distributed architecture, in which the various functions of the dialogue component are realized in different modules, as shown in Figure 14. The task module represents the task structure of an application, the belief module represents the interpretation of utterances in the current dialogue context, and the dialogue module, which contains rules for dialogue behavior, deals with predictions for the next user

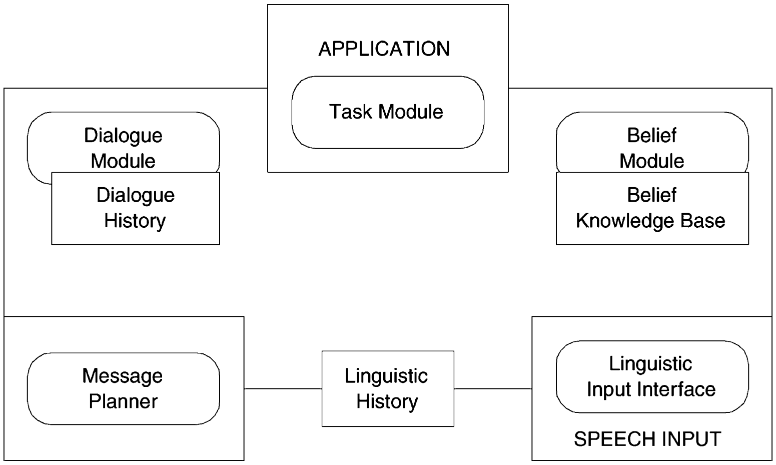


Fig. 14. The SUNDIAL architecture.

utterance and strategies for how the dialogue should continue. The following dialogue can be used to illustrate this method:

System1: *Hello Sundial reservation system. Can I help you?*
User1: *I'd like a ticket from London to Paris.*
System2: *London to Paris. When do you plan to leave?*

At the point in the dialogue where the user has just produced User1, the Belief Module has recorded that the items Departure City and Arrival City were recognized with average recognition scores. The next items on the agenda for the task module are to request the Departure Date and Departure Time. Given these circumstances, the system could choose from a range of possibilities for continuing the dialogue:

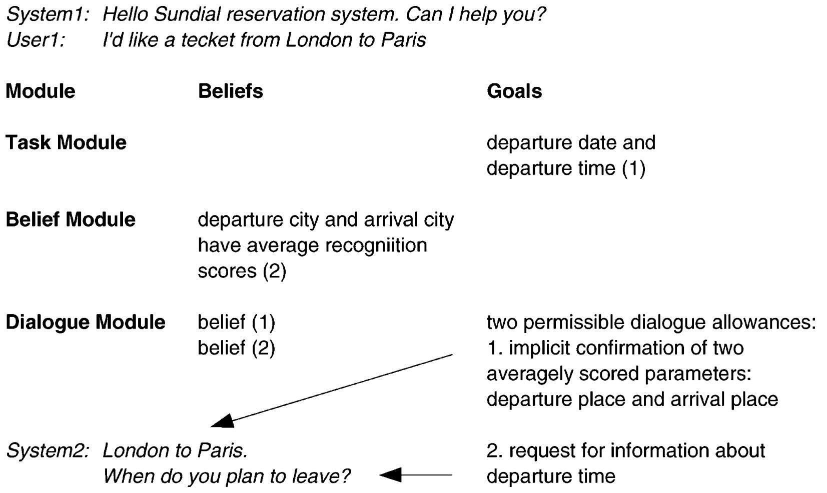
(1) explicit confirmation of departure city (*Did you say you want to fly from London?*);
(2) explicit confirmation of departure city (*Did you say you want to fly to Paris?*);
(3) explicit confirmation of arrival city (*Did you say you want to fly from London?*);
(4) request for departure date (*On which date do you wish to fly?*);
(5) request for departure time (*At what time do you wish to depart?*).

There are also many more possibilities involving further combinations of these items.

The solution that was adopted in the current example was S_2: *London to Paris. When do you plan to leave?* Here two possible dialogue allowances are combined: an implicit (rather than an explicit) request for confirmation, as the recognition score for the two parameters indicated reasonably reliable recognition, and a request for information about the departure time. Figure 15 presents a snapshot of the system at this point in the dialogue.

The processes that underlie this behavior are based on mappings from the system's belief state to the dialogue level [Heisterkamp and McGlashan 1996]. The user's utterance is analyzed semantically, yielding a surface semantic description in the SIL representation (see Section 4.2.2.1), and then interpreted in the context of the current task to provide a task-level interpretation. In this example, two new items are added to the system's contextual model: the departure place and the arrival place. The contextual functions of these semantic items derived from the user's utterance result in some change in the system's contextual model and are represented as follows:

—`new_for_system(goalcity:paris).`
—`new_for_system(sourcecity:london).`



System1: *Hello Sundial reservation system. Can I help you?*
User1: *I'd like a tecket from London to Paris*

Module	Beliefs	Goals
Task Module		departure date and departure time (1)
Belief Module	departure city and arrival city have average recogniition scores (2)	
Dialogue Module	belief (1) belief (2)	two permissible dialogue allowances: 1. implicit confirmation of two averagely scored parameters: departure place and arrival place 2. request for information about departure time

System2: *London to Paris.*
When do you plan to leave?

Fig. 15. Snapshot of the dialogue state.

Table III. Contextual Functions and Goals

`new_for_system(X)`	`confirm(X) or specify([X,Y])`
`repeated_by_user(X)`	`cancel 'confirm' goals for X`
`inferred_by_system(X)`	`introduce goal confirm(X)`
`modified_by_user(X)`	`introduce repair goal confirm(X)`
`negated_by_user(X)`	`repair(X)`

The next level of processing involves evaluating the contextual functions to see whether they solve a goal, modify a goal, or introduce a new goal. The complete list of contextual functions used and their associated goals are shown in Table III. The dialogue goals form a conflict set over which the dialogue strategy operates to determine the best dialogue continuation. The goals are grouped into classes: initiatives, responses, and evaluations. Generally, evaluations are given a higher priority than reactions, and reactions have precedence over initiatives, with the result than confirmations and answers to the user's questions will be resolved earlier than questions to be asked by the system.

Combining goals, as in this example, where the system provides both an implicit confirmation of two values as well as a further question within the same utterance (*London to Paris, when do you plan to leave?*), depends on the current state of the dialogue. The default setting is for any number of reactions (except confirm goals for inferred items) and one initiative to be combined, thus enabling the dialogue to proceed more quickly. However, a metastrategy of degradation and recovery, in which the dialogue goals are organized hierarchically from more open to more restricted interaction, as shown in Figure 16 [Giachin and McGlashan 1997], is used for situations when problems arise and the system has to focus on repair, and also to return to more flexible behavior as dialogue processing improves. If problems occur in a dialogue, the confirmation

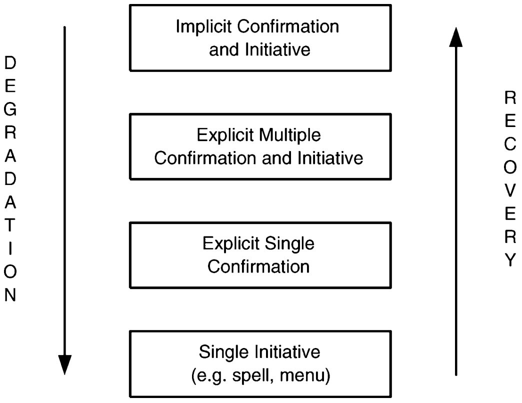

DEGRADATION
Implicit Confirmation and Initiative
Explicit Multiple Confirmation and Initiative
Explicit Single Confirmation
Single Initiative (e.g. spell, menu)
RECOVERY

Fig. 16. Metastrategy of degradation and recovery.

strategy changes from implicit through single explicit confirmation, moving at the lowest level to the use of spelling or menus. Once the problems are been resolved, the opposite sequence comes into operation.

The approach adopted in the SUNDIAL project overcomes the objections encountered in a state transition network approach in which the range of possible dialogue transitions would be extremely large at each node if flexible dialogue behavior were permitted that accounted for all the required circumstances, such as the degree of confidence in the recognition of items in the user's input. Thus, instead of a global dialogue control structure in which the dialogue paths are determined in advance, this approach adopts a local management strategy in which dialogue control is determined on a turn-by-turn basis depending on the information available in the various modules of the dialogue component. The focus on system beliefs and intentions avoids the difficulties of identifying the user's goals and attentions that are associated with plan-based approaches. This approach also offers a more generic architecture which in principle can be more easily ported to other applications, whereas in the state transition network approach each new application has to be constructed *ab initio*.

5.3.5. Dialogue as Rational Interaction.

The "Rational Interaction" approach to dialogue views communication as a special case of intelligent behavior. The following simple example illustrates this approach [Sadek and de Mori 1998]:

User: *Is server* 36–68–02–22 *operated by Meteo France?*
System: *(answers 'yes' or 'no').*

In order to make a decision as to how to answer the user's query, the system engages in the following sequence of actions:

(1) system infers the intention of the user to know if p (p = server 36–68–02–22 is operated by Meteo France);
(2) system adopts the intention that the user eventually comes to know if p;
(3) system adopts the intention of informing the user that p or of informing the user that *not* p.

This chain of reasoning may appear at first glance to be a case of "overkill" with regard to this simple example. However, the rationale for this approach becomes clear if the answer to the query is *no*, as in order to cooperatively address the user's intention, the system would have to plan to supply additional information, for example, that the server is operated by some other operator, or that it is some other server that is operated by Meteo France. This plan would involve reasoning about the relevance of the information, that is, what the user needs to know as well as what the user does not need to be told. Answering *yes* could also involve a chain of reasoning, for example, deciding whether the user is authorised to know p or inferring the consequences of telling the user that p.

In this approach dialogue structure emerges dynamically as a consequence of principles of rational cooperative interaction. Processes of dialogue can be explained in terms of the plans, goals, and intentions of the agents involved in the dialogue. Plans themselves are not predetermined schemas of action but are derived deductively from rationality principles. To take a simple example: since dialogue is a joint activity between two agents, each agent in the dialogue has a commitment to being understood. This commitment explains and motivates the need for confirmations and requests for clarification in dialogue [Cohen 1994]. While agents normally have the goal of behaving cooperatively in dialogue, an agent does not necessarily have to adopt another agent's goals, if there is good reason not to. For example, an agent should not supply information that is confidential or assist in actions that it knows to be illegal or harmful. In other words, an agent has to attempt to achieve a rational balance between its own mental attitudes and those of other agents and between these mental attitudes and desired plans of action.

The theoretical framework for rational agency is based on a set of logical axioms that formalize basic principles of rational action and cooperative communication. The foundations for this framework, developed by Cohen and Levesque [1990], were extended by Sadek and implemented as the basis of the dialogue management component of the ARTEMIS system. ARTEMIS is an agent technology developed at France Telecom-CNET as a generic framework for specifying and implementing intelligent dialogue agents [Sadek and de Mori 1998]. The AGS application, which was developed using ARTEMIS, provides information in the areas of employment and weather forecasts [Sadek et al. 1997]. In addition to the usual components of a spoken dialogue system, the dialogue manager of ARTEMIS applications includes a "rational unit," which supports reasoning about knowledge and actions and enables the dialogue agent to produce rationally motivated plans of action, including communicative acts, in response to the user's utterances. The essential elements of the rational unit are a number of principles of rationality and cooperation.

The rationality principles describe an agent's reasoning about actions, beliefs, and plans to modify the world, for example, through actions, including communicative acts that change another agent's mental state. An agent's intentions are defined in terms of the agent's beliefs about the world, as well as its goals and commitments. The principles are formalized as a set of logical axioms. Some examples will give a flavor of the formalism and how it is applied to dialogue agency.

One of the properties of a rational agent is consistency of beliefs. This property is modeled in terms of mental attitudes such as belief (B) and choice (C), as follows:

$$B(i, \phi) \Rightarrow \neg(i, \neg\phi).$$

That is, if agent i believes formula schema ϕ, then he or she will not believe that ϕ is not true. The logical model for choice specifies that an agent chooses the logical consequences of its goals:

$$\models (C(i, \phi) \wedge B(i, \phi \Rightarrow \psi)) \Rightarrow C(i, \psi).$$

That is, it is a valid formula that, if agent i desires that ϕ should be true and believes that ψ is a consequence of ϕ, then he or she will adopt the desire that ψ be true. Within the world of action, there is a need for an action model that specifies the rational effect (*RE*) of an action, that is, the reasons for which an action is performed. Also needed are the action's feasibility preconditions (*FP*), that is, the conditions that have to be true for the action to be feasible. The following simple example illustrates the act of informing:

$$\begin{aligned}&\langle i,\ Inform(j, \phi)\rangle\\ &\qquad FP\colon B(i, \phi) \wedge \neg B(i, B(j, \phi))\\ &\qquad RE\colon B(j, \phi).\end{aligned}$$

In other words, for i to inform j about ϕ, it must be the case that i believes ϕ and that i believes that j does not already know ϕ. The first part of the feasibility precondition models the sincerity condition that is an integral element in cooperative communication, while the second part models principles of quantity and relation, that is, that an agent should not tell another agent something that they already know. The rational effect of this communicative act is that agent j comes to know ϕ.

The cooperation principles express the motivation for an agent to behave cooperatively with respect to another agent. Agents in dialogue accept a number of minimal commitments, for example, to participate actively in the conversation, to attempt to understand the other agent's concerns, and to generate answers to these concerns that are cooperative. For example: there is a commitment to ensure mutual understanding, which can involve providing confirmations or making requests for clarification, as well as taking care that the other agent does not have erroneous beliefs. Cooperation also involves adopting the other agent's intentions, provided that this does not conflict with the first agent's own intentions. These principles of cooperation have been formalized in a similar way to the rationality principles.

The view that dialogue is a special case of rational behavior brings several advantages. Given that dialogue involves a joint commitment to mutual understanding, there is a motivation for agents to make their intentions clear through confirmations, clarifications, repairs, and elaborations. Although these behaviors are included in other approaches, there is no theoretical motivation for their inclusion. The theory also accounts for different contexts of interaction and explains why an agent might provide more information than is required by the other agent's query. For example: if a user asks for an address, the system might also provide a telephone number, if one is available. However, this additional information should not be implemented as an automatically generated response schema but rather as something to be determined within a particular context of interaction on the basis of the rationality principles. Finally, the theory provides a basis for more advanced dialogues, for example, those involving negotiation rather than simple information retrieval, where various types of cooperative and corrective responses may be required.

5.3.6. Advantages of Agent-Based Systems. As mentioned earlier, finite state-based and frame-based dialogue control methods are not suitable for more complex dialogues that involve more than the elicitation of a number of predetermined parameters to enable some information to be retrieved from a database or a simple transaction to be performed. More complex technologies are required for dialogue systems that support collaborative problem solving. Such dialogues involve negotiation of a task in which both system and user ask questions, request clarifications, make corrections and suggestions, and change the topic. In a mixed-initiative dialogue such as this, the dialogue agent has to be able to maintain and reason over explicit models of the task at hand, of the current dialogue state, and of its own and the other agent's beliefs and intentions. The agent-based dialogue systems described in this section, although mainly still at the research laboratory stage, illustrate a number of different approaches to the long-term goal of enabling conversational interaction with computers.

5.3.7. Disadvantages of Agent-Based Systems. The main disadvantage of agent-based dialogue control is that it requires much more complex resources and processing than the simpler dialogue control methods. In order to engage in a mixed-initiative dialogue involving negotiation and collaboration, a system requires more sophisticated natural language capabilities. Whereas simple pattern-matching and concept-spotting techniques are sufficient for finite state and frame-based systems, a deeper semantic representation is required to interpret the user's input in more open-ended mixed-initiative dialogues.

Integration of dialogue control with domain and task knowledge is a further challenge for agent-based systems. It is possible to handcraft domain-specific and task-specific knowledge into a dialogue system and then to use simpler dialogue control methods. However, this raises the issue of reusability as systems have to be completely redesigned when porting to new domains. A more satisfactory solution is to develop a generic domain-independent dialogue management component that can be easily adapted to new tasks. Essentially this is the solution adopted in the agent-based systems described in this section. So, for example, in the theorem-based approach, while the domain processor is application-specific, the general reasoning component and the knowledge component are domain-independent and incorporate general mechanisms for reasoning with the knowledge contained in the domain processor (see Section 4.4.2). The same applies to systems based on planning (Section 5.3.2) and rational agency (Section 5.3.5). In the TRIPS project, the successor to the TRAINS project (Section 5.3.3) an architecture has been developed including an abstract problem-solving model that supports the underlying structure for a collaborative task-based dialogue in terms of key concepts such as objectives, solutions, resources, and situations. Porting to a new domain and task is a matter of specifying mappings from this model to operations in the new domain [Allen et al. 2000].

Many of the techniques required to support agent-based dialogue control, such as intention recognition and reasoning, are computationally intensive. Finding ways of implementing these techniques to enable real-time performance and to support open-ended, mixed-initiative dialogues remains a challenge for researchers in spoken dialogue.

5.4. Summary

With such a number of different approaches to dialogue management, it is reasonable to ask which approach is most appropriate for a particular application. Conversational agents that incorporate principles of rationality and cooperation would seem to be the obvious choice, as they come closest to modeling human conversational competence. Certainly, for applications that involve cooperative problem solving with negotiated solutions, the simpler types of dialogue control are not sufficient. On the other hand, for simple applications and for constrained subtasks within some applications, more basic techniques such as finite state and template-based control may be appropriate.

Finite state networks are suitable for simple, well-structured tasks that can be decomposed into clearly defined subtasks. They are also suitable for tasks involving sequential form filling. A finite state-based system will be system-driven with a predefined dialogue path from which the user cannot deviate. Ideally the user's responses should be short, in the form of single words or simple phrases, and a major task in the design of such a system requires careful wording of the system prompts to constrain the user to answer in this way. When designed using a state transition diagram, these systems have a clear and intuitive semantics. The disadvantage of finite state systems is that they are inflexible and cannot be easily designed to permit the user to make a correction of some previously elicited value or to change easily from one subtask to another. Some of these deficiencies have been addressed by incorporating additional functionalities such as natural language processing and frames into the basic model [McTear et al. 2000]. However, these additions tend to obscure the semantics of the system and increase the number of alternative paths through the dialogue, with the potential of leading to combinatorial explosion. For these reasons finite state systems are best used for simple tasks involving the elicitation of a small number of items from the user, or for well-defined subtasks within a larger application, such as eliciting a date or time.

Frame-based systems permit a greater degree of flexibility, as the user can provide more information than required by the system's question, and in addition does not need to supply all the information at one time if this is not possible. The system keeps track of what information is required and asks its questions accordingly. The design of a frame-based system is declarative, as in a production system, thus providing a clear semantics. However, frame-based systems are restricted to basic information retrieval tasks with no facility for negotiation of the information. Some of the dialogue behaviors that a frame-based system might display, such as clarifying old information before asking for new values, are hard-coded into the system's control structure. In a system based on conversational agency these behaviors would be more flexible and would be based on a process of explicit deliberation by the system. Thus, for applications involving the elicitation of a fixed set of information from the user and the retrieval by the system of a clearly defined set of information in return, a frame-based system provides a suitable solution with greater flexibility than a finite state-based system. The Philips train timetable system is a good example of such an application. However, an application that involved consideration of constraints imposed by the user's goals, such as the need to make particular connections or to be able to change goals during the course of an interaction, would be beyond the scope of a frame-based system.

There is a wide range of agent-based approaches. These are usually motivated by particular theories of dialogue. The Circuit-Fix-It Shop system views problem solving as theorem proving. Dialogue control evolves dynamically through the mechanism of interruptible theorem proving, which is used to deal with missing axioms and user requests for clarification and help. Axioms that describe aspects of the user's domain-specific knowledge are used effectively to enable the system to engage in a limited amount of user modeling. Other approaches, such as those adopted in TRAINS, SUNDIAL, and Traum's conversational agency, are motivated more by linguistic theories of dialogue, where the focus is on a cooperative dialogue strategy that evolves dynamically over the course of the interaction. In these systems goals, beliefs, intentions, and obligations form a conflict set that is resolved according to a particular conversational metastrategy. Finally, in the rational agency approach, communication is viewed as a special case of rational behavior and dialogue control is determined in terms of axioms that encode principles of rational cooperative behavior.

While there have been no comprehensive studies to date of the costs and benefits of these different dialogue management approaches, some of the methods that have been developed for the design and evaluation of spoken dialogue systems address these issues in part. These will be reviewed in the next section, while Section 7 will examine some dialogue development tools that are available to build finite state and frame-based systems.

6. SPECIFYING, DESIGNING, AND EVALUATING A SPOKEN DIALOGUE SYSTEM

Developing a spoken dialogue system can be viewed as a special case of software engineering, with its own methods and evaluation criteria that have evolved over the past few years. A recent EU project, DISC (Spoken Language Dialogue Systems and Components), is concerned with specifying a best-practice methodology for the development and evaluation of spoken dialogue systems [Dybkjær et al. 1997]. Various sets of guidelines and standards have emerged as a result of research projects such as the Danish Dialogue Project [Bernsen et al. 1998], the EU-funded EAGLES project on standards for spoken language systems [Gibbon et al. 1997], and projects funded in the US under the ARPA initiatives on spoken language systems [Hirschman 1995]. The main trends in this work are reviewed in this section, looking first at the methods employed to support the specification, design, and development of spoken dialogue systems, and then at methods of evaluation that have been used.

6.1. Development Methodologies

Developing a spoken dialogue system involves deciding on the tasks that the system has to perform in order to solve a problem interactively with a human user; specifying a dialogue structure that will support the performance of the task; determining the recognition vocabularies and language structures that will be involved; and designing and implementing a solution that meets these criteria.

Various methods are in common use for establishing system requirements. These include: literature research, interviews with users to elicit the information required to construct the domain and task models; field-study observations or recordings of humans performing the tasks; field experiments, in which some parameters of the task are simulated; full-scale simulations; and rapid prototyping. In order to illustrate the issues involved, the two most commonly applied methods—design based on an analysis of human-human dialogues, and design based on simulations—will be described, followed by a discussion of usability issues and of design guidelines and standards.

6.1.1. Design Based on the Analysis of Human-Human Dialogues. Human-human dialogues provide an insight into how humans accomplish task-oriented dialogues. Considerable effort has gone into collecting corpora of relevant dialogues, many of which are publicly available, such as the TRAINS corpora and the CSLU corpora. Analysis of the TRAINS corpora can provide information about the structure of the dialogues in support of conversational modeling—for example, whether task-oriented dialogues consist mainly of a single topic—and about the range of vocabulary and language structures involved. The CSLU corpora, on the other hand, are focused mainly on modeling accents and multilingual pronunciations.

Analysis of natural dialogues may also pinpoint some aspects of how humans interact with software such as email and calendar applications when they are using a speech-based rather than a graphical user interface. In the SpeechActs projects at Sun Microsystems Laboratories, predesign studies are used before dialogue design to help the designer view the task from the user's perspective and to develop a feel for the style of interaction [Yankelovich n.d.]. One of the findings was that users of the calendar application typically used relative dates such as *tomorrow* or *next Monday*, whereas absolute dates would be used in the version provided in the graphical user interface. The organization of information, such as the numbering of messages in Sun's Mail Tool GUI, gave rise to confusion in the spoken language interface as it became difficult to keep track of which messages were new and to refer back easily to previously read messages [Yankelovich et al. 1995]. Thus it was concluded that users of a speech user interface (SUI) employ a different set of mental abilities compared to when they use a graphical user interface (GUI). For this reason it was recommended that methods need to be developed to cope for the lack of visual cues when interacting with software applications over a telephone line. The predesign studies had an important bearing on issues such as these as well as for the design of prompts, the selection of verification strategies, and the provision of immediate feedback.

6.1.2 Design Based on Simulations: Wizard of Oz and "System in the Loop." Although the analysis of human-human dialogues can provide useful information to support the design of spoken dialogue systems, the main drawback of this approach is that it is not possible to generalize from unrestricted human-human dialogue to the more restricted human-computer dialogues that can be supported by current technology. Current systems are restricted by limited speech recognition capabilities, limited vocabulary and grammatical coverage, and limited ability to tolerate and recover from error. To investigate how humans might talk to a more restricted dialogue partner, such as a computer system in a situation where no such system presently exists, some sort of simulation of the system is required.

The Wizard-of-Oz (WOZ) method is commonly used to investigate how humans might interact with a computer system [Fraser and Gilbert 1991]. In this method a human simulates the role of the computer, providing answers using a synthesized voice, and the user is made to believe that he or she is interacting with a computer. The situation is controlled with scenarios, in which the user has to find out one or more pieces of information from the system, for example, a flight arrival time and the arrival terminal. The use of a series of carefully designed WOZ simulated systems enables designs to be developed iteratively and evaluation to be carried out before significant resources have been invested in system building [Gibbon et al. 1997]. One of the greatest difficulties facing the WOZ method is that it is difficult for a human experimenter to behave exactly as a computer would, and to anticipate the sorts of recognition and understanding problems that might occur in the real system.

To overcome this disadvantage, the "System in the Loop" method may be used. In this case, a system with limited functionality is used to collect data. For example: the system might incorporate on the first cycle speech recognition and speech understanding modules, but the main dialogue management component may still be missing. On successive cycles additional components can be added and the functionality of the system increased, thus permitting more data to be collected. It is also possible to combine this method with the WOZ method, in which the human wizard simulates those parts of the system that have not yet been implemented.

6.1.2. Usability Analysis. As with any other software, the success of a spoken dialogue system does not depend solely on the functionality and performance of the software but also on its usability and acceptance by the users for whom it is intended.

One aspect of usability is to determine the costs and benefits of the proposed system. Lennig et al. [1995] described the development of a system that aimed to automate the handling of some directory assistance (DA) calls in Bell Canada. One of the initial investigations involved determining where potential savings could be made. Since the average operator work time per call was found to be approximately 25 seconds, and the cost to companies in the US of providing directory assistance was estimated to be over $1.5B, a reduction of 1 second in the work time per call would represent savings of over $60M per year. Operator acceptance was another factor that was investigated in this study. Operators were generally positive and particularly welcomed the fact that with the automatic system they did not have to continually repeat the same information. Using the system was easier on their voice and also required less keying, thus avoiding the problems of repetitive strain injury.

Similar findings emerged in a part-automated directory enquiries system developed by Vocalis for Telia TeleRespons, Sweden's leading network services provider [Peckham n.d.]. A major concern was how to reduce the running costs of directory enquiries without compromising customer satisfaction. The solution was a semiautomated system using a combination of voice response and speech recognition technology. The main tasks of an operator handling a directory enquiry call are identified and those parts that can be handled automatically are specified. Voice processing technology is used at the beginning and end of calls—to greet the caller and to release the requested number. The search for the number is handled by the human operator. Word spotting allows the speech recognition component to recognize key words in the midst of extraneous words and sounds, while "talkover" allows the caller to speak over the system output if they do not wish to wait for the machine to finish before they begin speaking. The commercial benefits of the system include increased staff productivity and substantial cost savings. From the technical perspective, the system achieves a speech recognition accuracy level of 97%, while Telia TeleRespons benefits from an 8% increase in efficiency and savings of millions of pounds each year. Research from Telia TeleRespons shows that over 90% of people are pleased to use the system and that an additional 6–7% actually prefer it. All the significant factors, including market conditions, pricing, the role of the operators, and the views of the unions, suppliers, and customers, were analyzed at the outset before the technical requirements of the system were considered.

6.1.3. Requirement Specification. Following the analysis of requirements for a spoken dialogue system based on one or more of the methods described in the preceding paragraphs, a formal requirements specification can be produced. The most elaborate approach would appear to be that employed in the Danish Dialogue Project in which two sets of documents—a Design Space Development (DSD) and a Design Rationale (DR)—are produced [Bernsen 1993]. A DSD document (or frame) represents the design space structure and designer commitments at a given point during system design, so that a series of DSDs provide a series of snapshots of the evolving design process. A DSD contains information about general constraints and criteria as well as the application of these constraints and criteria to the system under development in the Danish Dialogue

Table IV. Contextual Functions and Goals

A. General constraints and criteria	**B. Application of constraints and criteria to the artefact within the design space**
Overall design goal: Spoken language dialogue system prototype operating via the telephone and capable of replacing a human operator. *Realism criteria:* The artefact should be preferable to current technological alternatives. The system should run on machines which could be purchased by a travel agency. *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.	*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. *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). **C. Hypothetical issues** Is a vocabulary of 500 words sufficient to capture the sublanguage vocabulary needed in the task domain?

Project. Table IV shows some sample entries from a DSD (from [Dybkjær et al. 1996]). A DR frame represents the reasoning about a particular design problem. An example given in Dybkjær et al. [1996] describes a feature which had not been taken into account in the original specification—that users were not able to get the price of the tickets they had reserved. The DR contains information concerning the justification for the original specification, a list of possible options, the resolution adopted, and comments. In this way the evolving design and its rationale are comprehensively documented.

In addition to requirement specification documents such as these, the EAGLES handbook [Gibbon et al. 1997] recommends a formal and explicit description of the proposed dialogue. One way to do this would be to represent the dialogue flow as a flowchart, state transition network, or dialogue grammar, in which all reachable states in the dialogue are specified, along with information on what actions should be performed in each state and how to decide which state should be the next. Tools exist for displaying this type of information graphically. For example, in the Danish Dialogue Project DDL (Dialogue Description Language), a graphical language for describing state transition diagrams for event-driven systems is used to provide a formal specification of spoken dialogue systems.

6.1.4. Design Guidelines. The theoretical basis for much of the work on design guidelines comes from a theory of cooperative conversation, developed by the philosopher of language, Grice [1975]. Grice identified a number of maxims underlying cooperative conversation concerning quantity, quality, relation, and manner of communication. For example: the quantity maxim stated that a speaker should be as informative as required, but not more informative than required; the quality maxim related to the truth of a conversational contribution, and the relevance maxim to its relevance. Some commonly used evaluation metrics, such as Contextual Appropriateness (Section 6.2.2), as well as the usability guidelines developed in the Danish Dialogue Project (see next section) are based loosely on Grice's work.

6.1.4.1. Guidelines in the Danish Dialogue Project. Gricean maxims have been developed and extended into a set of usability guidelines in the Danish Dialogue Project [Bernsen et al. 1996]. A first set of the guidelines was developed on the basis of analysis of 120 examples of user-system interaction problems identified in a corpus of dialogues from the Wizard-of-Oz (WOZ) simulations of the Danish dialogue system. The guidelines were subsequently refined and consolidated into a tool called DET (Dialogue Evaluation Tool), which can be used to support the design of cooperative dialogue systems and as a tool for diagnostic evaluation [Dybkjær et al. 1997]. DET consists of 22 guidelines grouped under seven different aspects of dialogue, such as informativeness and partner symmetry, and divided into generic (GG) and specific guidelines (SG). The following are some examples (those marked with * are based on Grice):

Informativeness

GG1:. *Make your contribution as informative as is required (for the current purposes of the exchange).

SG1:. Be fully explicit in communicating to users the commitments they have made.

Partner Symmetry

GG10:. Inform the dialogue partners of important nonnormal characteristics that they should take into account in order to behave cooperatively in dialogue. Ensure the feasibility of what is required of them.

SG4:. Provide clear and comprehensible communication of what the system can and cannot do.

(Source: [Dybkjær et al. 1997]).

Using the guidelines for evaluation involves analyzing transcripts of dialogues to identify instances of violations of the guidelines, which are marked up in the transcripts. The violations can then be examined in greater detail, disagreements between analyzers resolved, and recommendations developed for enhancing cooperativity in the dialogue system. The generality of the guidelines has been explored by applying them successfully as a dialogue design guide to part of a corpus from the Sundial project [Dybkjær et al. 1997].

6.1.4.2. The EAGLES Guidelines. In the EAGLES handbook a series of recommendations have been proposed to support the design of spoken dialogue systems. These guidelines include recommendations for the design of interactive voice response (IVR) systems and for the design of prompts. The following is a summary of the recommendations for the design of dialogue systems [Gibbon et al. 1997]:

(1) Data collection:
  - —Study of recordings of human-human interaction in a situation similar to the one in which the system will be used.
  - —Wizard-of-Oz simulations.
  - —Transcription of the dialogues.
(2) Specification, design, and implementation of a first version (X) of the dialogue system.
(3) Tests:
  - —Laboratory tests using corpora recorded in Wizard-of-Oz simulations, and then with laboratory staff simulating users, recording new data.
  - —Field tests with real users, recording new corpora.
(4) Tune the system by iteratively modifying, then testing it.
(5) Design and implement an X + 1 version of the system, integrating new technologies.
(6) Tests (as in step 3).
(7) Return to step 4 unless the system is deemed to be complete.

Specific recommendations concerning the dialogue model and the vocabulary of the system are included in the following additional guidelines:

(1) Conduct a dialogue act analysis of the dialogues collected in the corpora, paying special attention to the conditions that must be satisfied in order to proceed from one dialogue state to the next.
(2) Describe the dialogue state transitions using some formally explicit apparatus (such as a flowchart or formal specification language).
(3) Use the data to identify the total lexicon required, then divide it into sublexicons, where each sublexicon is associated with a dialogue act.
(4) Use the data to identify a covering grammar, then divide it into subgrammars, where each subgrammar is associated with a dialogue act.

6.2. Evaluation

Most of the methods used to support the specification and design of spoken dialogue systems—such as corpus analysis, WOZ, and system-in-the-loop—can also be used to collect data for evaluation. This section will focus on the metrics that are employed rather than on the methods of data collection.

Evaluation of spoken dialogue systems can involve either evaluation of the individual components (*glass box evaluation*), or evaluation of the system as a whole (*black box evaluation*). Evaluation of individual components, with measures such as word accuracy and sentence accuracy, have been employed for some time to measure the performance of spoken language systems under the ARPA initiatives [Hirschman 1995]. It is only more recently that measures have been developed for spoken dialogue systems as a whole. Both types of evaluation have been described in some detail in a review paper by Baggia [1996], from which much of the material in this section is derived. See also Smith [1997] and Smith and Gordon [1997] for a comparable set of evaluation methods.

6.2.1. Evaluation of Individual Components.

Evaluation of individual components is generally based on the concept of a reference answer, which determines the desired output of the component to be compared with its actual output. Reference answers are easier to determine for components such as the speech recognizer and the language understanding component, but more difficult with the dialogue manager where the range of acceptable behaviors is greater. The most commonly used measure for speech recognizers is *Word accuracy (WA)*. *WA* accounts for errors at the word level, which include insertion (W_I), deletion (W_D), and substitution of words (W_S). *WA* is calculated as a percentage using the formula

$$WA = 100\left(1 - \frac{W_S + W_I + W_D}{W}\right)\%,$$

where W is the total number of words in the reference answer.

Sentence accuracy (*SA*) is a measure of the percentage of utterances in a corpus that have been completely and correctly recognized. In this case the recognized string of words is matched exactly with the words in the reference answer. *Sentence understanding rate* (*SU*), on the other hand, measures the rate of understood sentences in comparison with a reference meaning representation. An alternative measure of understanding is *concept accuracy* (*CA*), which measures the percentage of concepts that have been correctly understood. *CA* is similar to *WA*, as it measures errors at the concept level which include insertions, deletions, and substitutions. *Text understanding* (*TA*), a measure used in the Message Understanding Conferences, measures the amount of significant information that has been extracted from a text, using templates as the reference answers. Finally, an evaluation method has been developed in the ARPA Spoken Language System program to measure the correctness of database query responses by matching the actual responses with reference answers expressed as a set of minimal and maximal tuples [Hirschman 1995]. The correct answer must include at least the information in the minimal answer and no more information than is in the maximal answer. This measure is similar to some of the measures for dialogue success to be discussed below.

Some interesting results have emerged from evaluation studies using these measures. Hirschman [1995] reported that, in the various ARPA evaluations, the error rate for sentence understanding was much lower than that for sentence recognition (10.4% compared with 25.2%), indicating that it is easier to understand sentences than to recognize them and that sentence understanding, by using robust processing techniques, is able to compensate to some extent for errors produced by the speech recognition component. Similar results were reported by Boros et al. [1996] in a comparison of word accuracy and concept accuracy measures. Boros et al. found that it is possible to achieve perfect understanding with less than perfect recognition, but only when the misrecognitions affect semantically irrelevant words. When misrecognition affects parts of the utterance that are significant for understanding, *CA* may be lower than *WA*. Thus it is important to examine closely the relationships between different measures.

6.2.2. Evaluation of Spoken Dialogue Systems.

The performance of a spoken dialogue system can be measured in terms of the extent to which it achieves its task, the costs of achieving the task (for example, the time taken or number of turns required to complete the task), and measures of the quality of the interaction, such as the extent to which the system behaves cooperatively (see the EAGLES handbook [Gibbon et al. 1997] for a detailed account of the measures described below together with annotated examples).

A set of core metrics was identified in the SUNDIAL project that measures these aspects of dialogic interaction [Simpson and Fraser 1993].

Transaction success (*TS*) is similar to the ARPA measure of the correctness of database query responses discussed earlier, in that this metric measures how successful the system has been in providing the user with the requested information. TS was defined as a four-valued measure to account for cases of partial success as well as instances where the user's goal was not clearly identifiable or changed during the course of the interaction: S (succeed), SC (succeed with constraint relaxation), SN (succeed with no answer), and F (fail).

Number of turns is a measure of the duration of the dialogue in terms of the number of turns taken to complete the transaction. An alternative measure is the time taken to complete the transaction. These measures can be used in conjunction with different dialogue strategies to give an indication of the costs of the dialogue, which may be compared with other measures such as transaction success or user acceptance.

Correction rate (*CR*) is a measure of the proportion of turns in a dialogue that are concerned with correcting either the system's or the user's utterances, which may have been the result of speech recognition errors, errors in language understanding, or misconceptions. A dialogue that had a high degree of CR might be judged to have high costs in terms of user acceptability, as well as potentially high costs financially.

Contextual appropriateness (*CA*) is a measure of the extent to which the system provides appropriate responses. The metric can be divided into a number of values, such as: TF (total failure), AP (appropriate), IA (inappropriate), AI (appropriate/inappropriate), and IC (incomprehensible). With TF, the system fails to respond to the user. IA is used for responses that are inappropriate, defined usually in terms of Gricean maxims ([Grice 1975], see above). AI is used when the evaluator is in doubt, and IC when the content of an utterance cannot be interpreted.

The *Behavioral Coding Scheme* is a similar measure of the quality of responses produced in an interaction with a spoken dialogue system [Sutton et al. 1995]. Behavioral coding classifies the user's utterances to an automated questionnaire into 11 different types, some of which are illustrated in Table V. The Behavioral Coding Scheme has proven useful for the objective evaluation of user behavior, for evaluating the performance of the system, and as a basis for further refinement of the system.

A number of more qualitative measures have been developed, including a metric for evaluating dialogue strategies, such as strategies for recovering from errors [Danieli and Gerbino 1995]. *Implicit recovery* (*IR*), which can be compared with implicit verification as illustrated in Section 4.3.2, has been defined as the ability to overcome errors produced by the speech recognizer or parser and to rectify these implicitly. This strategy contrasts with an explicit strategy involving correction that can be measured using the correction rate (CR) metric. The following example illustrates implicit recovery

Table V. Categories from the Behavioral Coding Scheme

Code	Response Class	Description	Example
AA1	Adequate Answer 1	Answer is concise and responsive.	S: Have you ever been married? U: Yes.
AA2	Adequate Answer 2	Answer is usable but not concise.	S: Have you ever been married? U: No I haven't.
AA3	Adequate Answer 3	Answer is responsive but not usable.	S: Have you ever been married? U: Unfortunately.
IA1	Inadequate Answer 1	Answer does not appear to be responsive.	S: What is your sex, female or male? U: Neither.
IA2	Inadequate Answer 2	User says nothing at all.	S: What is your sex, female or male? U: <silence>

(values understood by the system in angular brackets):

User1: *I want to go from Roma to Milano in the morning.* <arrival-city = MILANO, departure-time=MORNING>

System1: *Sorry, where do you want to leave from?*

User2: *From Roma.* <departure-city = ROMA, cost-of-ticket?>

System2: *Do you want to go from Roma to Milano leaving in the morning?*

Although the user's first utterance contains the concepts that the system requires to retrieve the desired information, the departure city has not been recognized, so the system takes into account the concepts that have been correctly understood and asks for the concept that was not understood. The user's second utterance contains the required information but additional words have been inserted at the recognition level that are interpreted as a request for the cost of the ticket. As this concept is not relevant in the current context, it is disregarded by the system and the user is asked to confirm the correct concepts. In this case the *IR* score is 100% and the system has succeeded in spite of recognition and parsing errors but without having to engage in explicit correction.

In a study comparing explicit and implicit recovery strategies and using the other measures described earlier such as contextual appropriateness, Danieli and Gerbino [1995] found that the system that used an explicit recovery achieved greater robustness in terms of dealing with errors, although at the cost of longer transactions. It was also found that, as users became more familiar with the system, the recovery results for the system using implicit recovery improved substantially. Thus several aspects have to be considered and balanced when evaluating a dialogue system, including transaction success and dialogue duration, which measure the ability of the system to find the required information, and contextual appropriateness, which measures the quality of the dialogue. However, it is not possible using these measures to determine whether the higher transaction success of the system using the explicit recovery strategy was more critical to performance than the efficiency of the system using the implicit recovery strategy.

A recently developed tool for the evaluation of spoken dialogue systems, PARADISE (PARAdigm for Dialogue System Evaluation), addresses the limitations of the methods discussed so far by combining various performance measures such as transaction success, user satisfaction, and dialogue cost into a single performance evaluation function, and by enabling performance to be calculated for subdialogues as well as complete dialogues [Walker et al. 1997]. In this framework the overall goal of a dialogue system is viewed in terms of maximizing user satisfaction. This goal is subdivided into the subgoals of maximizing task success and minimizing costs. The latter is in turn subdivided into efficiency measures and qualitative measures. A brief overview of this framework is provided in the following paragraphs, although for more detail and a comprehensive set of illustrative examples, see Walker et al. [1997], citeyearwalker:2.

Transaction success is calculated by using an *attribute value matrix* (*AVM*) that represents the information to be exchanged between the system and the user in terms of a set of ordered pairs of attributes and their possible values. For example: **departure-city** might have values *Milano*, *Roma*, *Torino*, *Trento*, while **departure-range** might have the values *morning, evening*. The correct values for each attribute are determined by scenarios (for example, the user might be required to find a train that leaves from Torino to Milano in the evening). These values are referred to as the scenario *keys*, and these are plotted on a *confusion matrix* along with any incorrect values that occurred during the actual dialogue. This confusion matrix is used to calculate the kappa coefficient, κ [Carletta 1996], which indicates how well the system has performed a particular task within a given scenario, using the following formula:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)},$$

where $P(A)$ is the proportion of times that the AVMs for the actual dialogues agrees with the AVMs for the scenario keys, and $P(E)$ is the proportion of times that the AVMs for the dialogues and the keys are expected to agree by chance. Unlike other measures of transaction success and concept accuracy, κ takes into account the inherent complexity of the task by correcting for expected chance agreement.

Dialogue costs are measured in terms of cost measures c_i that can be applied as a function to any subdialogue. The information goals that an utterance contributes to are determined by using the AVM representation to tag the dialogue with the attributes for the task. In this way dialogue strategies used to achieve the task can be evaluated both in the dialogue as a whole as well as in subdialogues. Given a set of measures c_i the different measures are combined to determine their relative contribution to performance, using the formula

$$\mathit{Performance} = (\alpha * \mathcal{N}(\kappa)) - \sum_{i=1}^{n} w_i * \mathcal{N}(c_i),$$

in which α is a weight on κ, the cost functions c_i are weighted by w_i, and $\mathcal{N}$ is a $\mathcal{Z}$ score normalization function that is used to overcome the problem that the values of c_i are not on the same scale as κ and may also be calculated over varying scales. The weights for α and w_i are solved using multiple linear regression. Using this formula it is possible to calculate performance involving multiple dialogue strategies, including performance over subdialogues.

PARADISE is a framework for evaluating dialogue systems that incorporates and enhances previously used measures. It supports comparisons between dialogue strategies and separates the tasks to be achieved from how they are achieved in the dialogue. Performance can be calculated at any level of a dialogue, such as subtasks, and performance can be associated with different dialogue strategies. Furthermore, subjective as well as objective measures can be combined and their relative cost factors to overall performance can be specified. PARADISE has been used as a tool to evaluate a number of applications, including accessing train schedules and email as well as voice dialing and messaging [Walker et al. 1998; Kamm et al. 1999].

6.3. Summary

It can be seen that considerable attention has been devoted in recent years to the engineering aspects of spoken dialogue systems and to their specification, design, and evaluation, and that there is some degree of convergence on methodologies and frameworks, due in large part to concerted efforts in large-scale research and development projects sponsored by ARPA and the EU, and also due to the view of best practice in dialogue engineering as a specialization of best practice in software engineering, exemplified in projects such as DISC. A further factor is the increasing availability of large corpora that can be used both to support initial specification and design of systems and also as data for evaluation studies. It is salutary to conclude this section with a comment on the importance of customer satisfaction for the success of spoken dialogue systems in the marketplace:

> *From a commercial perspective, the success of a spoken dialogue system is only slightly related to technical matters.... 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, cited in Dybkjær [et al. 1997])

7. TOOLKITS FOR DEVELOPING SPOKEN DIALOGUE SYSTEMS

The development of a spoken dialogue system is a complex process involving the integration of the various component technologies described in Section 4. It would be a formidable task to build and integrate these components from scratch. Fortunately a number of toolkits and authoring environments have become available that support the construction of spoken dialogue systems, even for those who have no specialist knowledge of the component technologies such as speech recognition and natural language processing. The following are some of the dialogue development environments that are currently available:

—the Generic Dialogue System Platform (CPK, Denmark);
—GULAN—An Integrated System for Teaching Spoken Dialogue Systems Technology (CCT/KTH);
—the CSLU toolkit (Center for Spoken Language Understanding at the Oregon Graduate Institute of Science and Technology);
—CU Communicator system;
—the Nuance Developers' Toolkit (Nuance Communications);
—SpeechWorks;
—Natural Language Speech Assistant (NLSA) (Unisys Corporation);
—SpeechMania^{TM}: A Dialogue Application Development Toolkit (Philips Speech Processing);
—the REWARD Dialogue platform;
—Vocalis SpeechWare^{TM}.

The first three systems were developed mainly to support academic research and to support the teaching of spoken language technology. The CPK toolkit, developed at the Centre for PersonKommunikation at the University of Aalborg in Denmark, has been incorporated into the REWARD dialogue platform and is accompanied by a Web-based course. This material, which includes details of the development platform to be used for implementation, is currently not available publicly. GULAN, a system for teaching spoken dialogue technology, is under development at KTH (Stockholm) and at Linköping University and Uppsala University [Sjölander et al. 1998]. The system, which is currently in Swedish but due to be ported to English, is presently only runnable locally. The CSLU toolkit, to be described in greater detail below, is available free-of-charge under a license agreement for educational, research, personal, or evaluation purpose. The commercial systems are available under a range of license agreements. Some systems are available as evaluation versions and others can be obtained at a relatively low cost for academic purposes. Web sites with further information about these systems, including pricing, are listed in Appendix B.

A comprehensive description and evaluation of all these systems is beyond the scope of the current survey. To give a flavor of what is available, one academically oriented system, the CSLU toolkit, and one commercial system, the Philips SpeechMania^{TM} system, will be examined, followed by a brief outline of desirable features of spoken dialogue toolkits.

7.1. The CSLU Toolkit

The CSLU toolkit has been developed at the Center for Spoken Language

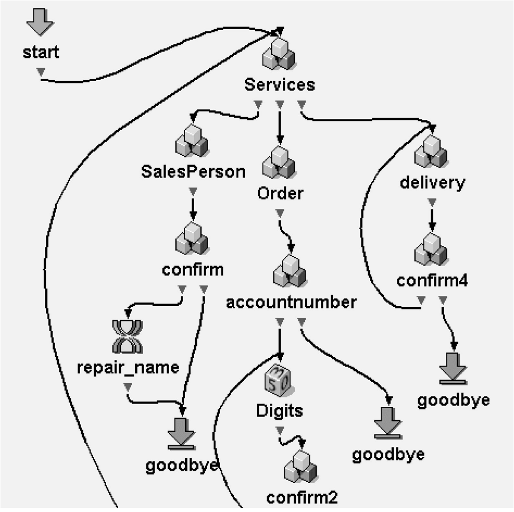


Fig. 17. Using RAD to simulate an auto attendant at a furniture store.

Understanding (CSLU) at the Oregon Graduate Institute of Science and Technology to support speech-related research and development activities [Sutton et al. 1998]. The toolkit includes core technologies for speech recognition and text-to-speech synthesis, as well as a graphically based authoring environment (RAD) for designing and implementing spoken dialogue systems. This section will focus only on RAD. Information about other components of the CSLU toolkit can be found at the CSLU web site (see Appendix B).

A major advantage of the RAD interface is that users are shielded from many of the complex specification processes involved in the construction of a spoken dialogue system. Building a dialogue system involves selecting and linking graphical dialogue objects into a finite-state dialogue model, which may include branching decisions, loops, jumps, and subdialogues, as illustrated in Figure 17. Each object can be used for functions such as generating prompts, recording and recognizing speech, and performing actions. As far as speech recognition is concerned, the input can be in the form of single words, for which a tree-based recognizer is used, or as phrases or sentences that are specified using a finite state grammar, which also enables key word spotting. There are additional built-in facilities for digit and alpha-digit recognition. The words specified for recognition at a given state are automatically translated by the system into a phonetic representation called Worldbet using built-in word models stored in dictionaries. Pronunciations can also be customised using the Worldbet symbols. It is also possible to implement dynamic recognition, in which case a list of words to be recognized is obtained from some external source, such as a Web page, and pronunciation models for the words are generated dynamically at run-time. Prompts can be specified in textual form and are output using the University of Edinburgh's Festival TTS (text-to-speech) system, or they can be prerecorded, and, with some additional effort, spliced together at run-time. The use of the subdialogue states permits a more modular dialogue design, as subtasks, such as eliciting an account number, can be implemented in a subdialogue that is potentially reusable. Repair dialogues are a special case of subdialogue. A default repair subdialogue is included that is activated if the recognition score for the user's input falls below a given threshold, but it is also relatively easy to design and implement customized repair subdialogues. There is also a special dialogue object for inserting pictures and sound files at appropriate places in the dialogue without the need for complex programming commands. The list-builder object simplifies the programming of a repetitive series of exchanges, such as questions, answers, and hints in an interactive learning programe, by allowing the programmer to specify lists of questions, answers, and hints in a simple dialogue box with the system looping through each of the alternatives either in serial or random order. A number of online tutorials, accompanied by simple illustrative examples of dialogue systems, provide an introduction to the basic functions of RAD.

Functions are provided in RAD for voice-based Web access. For example, a given URL can be accessed and the HTML document read and parsed, relevant strings can be identified, tags removed, and the required information output using text-to-speech. Although not documented in the current online tutorials, it is also relatively simple to develop an interface to databases and spreadsheets. Recently a natural language processing component has been developed that allows recognized strings to be parsed and relevant concepts to be extracted [Kaiser et al. 1999]. Finally, the toolkit includes an animated conversational agent (BALDI), developed at University of California at Santa Cruz, which presents visual speech through facial animation synchronized with synthesized or recorded speech.

RAD is currently being used effectively to provide interactive language learning for profoundly deaf children [Cole et al. 1999a] and to provide a practical introduction to spoken dialogue technology for undergraduate students [McTear 1999]. Plans are underway to develop multilingual versions of the toolkit [Cole et al. 1999b].

7.2. SpeechMania^{TM}

SpeechMania^{TM}, a product of Philips Speech Processing, is an application development environment to support the development of telephone-based spoken dialogue systems. The software allows people to talk with computers over the phone to access information services such as railway and flight timetables, bank statements, and stock exchange quotations, or to engage in transactions such as reserving a hotel room or reserving seats for a movie through a call center. The basic system architecture is shown in Figure 18.

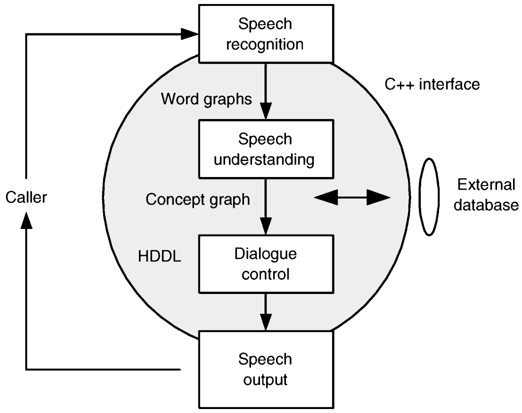


Fig. 18. The SpeechMania^{TM} architecture.

Processing is divided into modules for speech recognition, speech understanding, dialogue control, and speech output, with serial communication between the modules. Speech recognition, which is based on hidden Markov models with continuous mixture densities, is provided as part of the system in the form of acoustic models for particular languages such as American or British English, German, and Dutch. The other modules are specified using HDDL (High-level Dialogue Description Language), a dedicated declarative programming language for automatic enquiry systems. HDDL consists of a number of sections, of which two of the most important are Rules and Actions. The Rules section contains attributed context-free grammars for speech understanding (see Section 4.2.2.3 for an example). Dialogue control is specified in the Actions section using conditional actions (condactions) (see Section 5.2.1). For speech output all the required words and phrases are prerecorded and the appropriate segments are concatenated and replayed as specified by the dialogue control module. Finally, there is a transaction interface to external systems such as databases.

In addition to these modules, SpeechMania^{TM} includes a range of tools that support the development and evaluation of a spoken dialogue system. Applications can be tested offline using a text-based interface. The HDDL code can be checked using the HDDL parser, which in addition generates a list of words not present in the system's speech recognition lexicon and produces a list of the system prompts for recording using the Recording Station tool. A transcription tool enables the developer to optimize the system's speech recognition by comparing the words spoken by the user in a series of logged dialogues with the words recognised by the system. Dialogues can be marked up for subsequent statistical analysis, system evaluation, and system refinement

Table VI. Features of Spoken Dialogue Toolkits

Dialogue design	Visual programming Subdialogues
Prompts	Recording tools TTS
Simulation and testing	Design prototyping and simulation WOZ Offline testing Data capture
Natural language understanding	Grammar development
Multimodality	Graphics Facial animation
System training and tuning	Training the speech recognizer and other components
Interfaces	Databases, Web, APIs
Reusable components	Commonly used recognizers, grammars, subdialogues
Platform	Programming languages and environment

through training of the language model and the stochastic grammar.

7.3. Features of Spoken Dialogue Toolkits

Spoken dialogue toolkits can be compared and evaluated across a number of dimensions. Table VI presents a preliminary set of features that can be used. This list needs to be treated with some caution, however, as the presence or absence of a particular feature has to be considered in relation to the users and uses for which a given toolkit is intended. For example, in respect of ease of use, the CSLU toolkit, due to its graphical authoring environment, provides an excellent facility for students to quickly develop and test small dialogue systems. In contrast, developing even a simple system with SpeechMania^{TM} involves a relatively steep learning curve as the dialogue control and natural language understanding have to be programmed in HDDL. On the other hand, however, SpeechMania^{TM} provides a number of powerful tools to support the developer, including a fully functional telephone interface, that are not available in the CSLU toolkit. Thus when comparing the two toolkits it is important to consider their main purposes: the CSLU toolkit is designed mainly as an educational tool, whose graphical interface shields developers from the complexities involved in programming an interface using some sort of programming language. SpeechMania^{TM}, as well as the Nuance toolkit, in which the functions for state transitions and other state functions are programmed using C-code, are sophisticated development environments intended for commercial software developers.

Most of the toolkits listed above use some sort of visual programming to represent dialogue states and transitions. This facility is useful as long as the dialogues are simple enough to be implemented using finite state methods (see Section 5). To date SpeechMania^{TM} is the only toolkit that provides an alternative form of dialogue control that is programmed declaratively rather than visually.

Facilities for developing system prompts vary across the toolkits. In RAD the Festival TTS is closely integrated with the graphical authoring environment, with the result that prompts can be designed by simply typing in the text to be synthesized into a prompt dialogue box. There are also facilities for adjusting various parameters in the synthesized voice as well as recording prompts as wave files. In SpeechMania^{TM}, as in other toolkits such as NLSA, there is a sophisticated recording tool that is used to record and manipulate sections of sound files that can be concatenated at run time to provide system prompts.

Support at the design and testing stages is an important feature that is available in most toolkits. The CSLU toolkit supports rapid prototyping through its easy-to-use graphical authoring environment. There is also a facility for data capture that allows complete dialogues to be recorded and replayed. SpeechMania^{TM} also supports data capture but in addition the recorded items of user input can also be used by tools provided within the development environment to optimize speech recognition and understanding and to provide data for training and testing. Similar facilities are provided in the Nuance toolkit. The NLSA Dialogue Assistant not only provides a visual representation of the dialogue flow but also a simulation tool that enables Wizard-of-Oz testing of the viability of an application even before a speech recognition module has been installed. With the NLSA Dialogue Assistant the developer can simulate an application using actual phone lines, select system prompts and responses, and record caller responses, thus collecting data that can be used as a basis for subsequent redesign of the dialogue.

Most toolkits support some form of natural language understanding, generally with a semantic or concept-based grammar with terms that map closely on to domain-specific items such as entities in a database. Grammar development is generally not supported and it is the role of the developer to design the required grammar rules according to the formalism supported within the toolkit. One exception is the NLSA toolkit, which provides a tool Speech Assistant—to automate the process of creating grammars. Speech Assistant has a format similar to a spreadsheet. The words and phrases that constitute potential user utterances are entered into rows of the tables and for each set of related phrases a token is provided that represents the meaning of the responses. BNF grammars that are used by the speech recognition and natural language understanding components are compiled automatically from these tables.

Most spoken dialogue toolkits assume a telephone interface and so do not need to support pictures, animations, and other visual displays. The main exception is the CSLU toolkit, which, as described earlier, supports the display of pictures and also provides an animated talking head. As requirements emerge for multimodal spoken dialogue systems, as in public information kiosks, there will need to be greater support in toolkits for other modalities in addition to voice.

Each of the toolkits provides a number of additional tools and functions, for example, to customize and optimize speech recognition, and APIs are provided to enable a seamless interface to other applications. For example, the Festival TTS and BALDI animated face are launched automatically in RAD, and there are a number of supporting tools, such as SpeechView, to record, view, and manipulate waveforms. As expected, the commercially oriented toolkits such as Nuance and SpeechMania^{TM} have greater support for telephony functions, as these systems are most likely to be deployed with a telephone interface. There has also been more attention paid in these commercial toolkits to interfaces with external information sources such as large databases, although these facilities can also be provided with a little effort in RAD. Most of the toolkits provide their own speech recognition system, in some cases with barge-in facilities that allow the user to interrupt system prompts. NLSA is recognizer-independent, with support for a number of commercially available speech recognisers, so that developers can choose the recognizer that is best suited to their particular applications.

Reuse is an important issue for any software development environment. Each of the toolkits addresses this issue in some way. In RAD it is possible to create and save subdialogues and customized repair subdialogues for reuse in other applications. Similarly files of commonly used items, such as variants of words such as *yes* and *no*, can be loaded along with their pronunciations as required. Similar facilities exist in SpeechMania^{TM}. Other products take the issue of reuse still further. SpeechWorks provides DialogModules^{TM}, reusable objects containing prebuilt vocabularies and grammars as well as error-recovery routines that can be easily integrated into an application. Examples of these reusable objects include subdialogues for continuous digits, telephone numbers, zip codes, currency, and credit card numbers. The Nuance toolkit and NSLA have similar reusable components that can be imported into an application.

Most of the toolkits are available on several platforms such as Unix and versions of Windows such as 95, 98, and NT. The most commonly used languages are C, C++, TCL, HDDL, and Java. Many of the toolkits are compliant with speech API standards (SAPI) such as Microsoft SAPI and are customized for telephone API- (TAPI-) compliant telephony cards.

7.4. Summary

In this section a number of toolkits that support the development of spoken dialogue systems have been examined. Most of the toolkits provide fairly similar features with varying degrees of support for particular functionalities. The focus in this section has been mainly on the support provided in the toolkits for the dialogue management component. Naturally, the performance of individual components, in particular the speech recognizer, will have an important bearing on the acceptability of a dialogue system developed with a given toolkit, although this issue may become less relevant with the movement towards open architectures, in which developers can slot in recognizers and other components that suit their particular applications.

8. FUTURE DIRECTIONS

There are a number of ways in which spoken dialogue technology may develop over the next decade. As far as research is concerned, there are several initiatives aiming at conversational systems that support more natural mixed-initiative dialogues. The focus of much of this work is on working systems rather than on the development of the individual components. Thus, instead of concentrating, for example, on the development of a sophisticated natural language understanding component in isolation, research is likely to be directed toward the ways in which such a component can be integrated with the other components of a spoken dialogue system, and on how it can be deployed in real-world applications. Measurement of the performance of such components will be in terms of their contribution to the performance of the complete system.

Studies of human-human dialogue have provided useful insights into how more sophisticated dialogue systems might behave and have resulted in theories of cooperative interaction that inform the design and evaluation of interactive speech systems [Bernsen et al. 1998] as well as models of conversational agency [Traum 1996], which integrate AI work on planning with speech act theory. As speech recognition and natural language understanding become more robust, more sophisticated dialogue managers that have been developed in text-based systems as noted, for example, Carberry and Lambert [1999], will emerge. Jurafsky and Martin [2000] (Chapter 19) is a recent review of the theoretical background as well as recent developments in dialogue and conversational agency. Another trend is toward the use of statistical techniques in dialogue management. For example, dialogue sequences have been modeled as dialogue-act-*N*-grams in order to help predict upcoming dialogue acts [Nagata and Morimoto 1994]. Probabilistic methods are being used in conjunction with reinforcement learning algorithms to enable the automated learning of optimal dialogue strategies. Dialogue is modeled as a Markov decision process (MDP) and viewed as a trajectory in a state space determined by system actions and user responses. Given multiple action choices at each state, reinforcement learning is used to explore the choices systematically and to compute the best policy for action selection based on rewards associated with each state transition [Litman et al. 2000].

While most of this survey has been concerned with dialogue systems that provide a spoken language interface, there has been a recent development toward the integration of spoken language technology with other modalities [Cohen and Oviatt 1995]. In the TRAINS project, for example, the system displays a map of the area under discussion with the route being planned marked and highlighted. Some of the travel information systems, such as the ATIS (Air Travel Information System) in the United States and the EU Esprit MASK project involve multimodal interaction. For example, the MASK system is planned as a multimodal, multimedia service kiosk to be located in train stations, with the user being able to speak to the system as well as use a touch screen and keypad, while the system displays information to the user on a screen [Lamel et al. 1995]. The selection and coordination of different media in relation to different types of content to be displayed and the varying needs of the user and the task have been the subject of much research (see, for example, the papers in Maybury [1993]). Although this work is still in its infancy and many of the solutions adopted tend to be ad hoc and application-specific, there has been some progress toward a general theory of input and output modalities and of how speech might be integrated within a multimodal context [Bernsen 1994].

Another important application area is the World Wide Web. With the increasing integration of the Internet and domestic television, there is a potential for applications using spoken dialogue technology to perform services such as home shopping, or to control and program appliances around the home such as microwave ovens and VCRs. These needs are being addressed through VoiceXML (Voice eXtensible Markup Language)—an XML-based mark-up language for creating distributed voice applications that feature synthesized speech, digitized audio, recognition of spoken and DTMF key input, recording of spoken input, telephony, and mixed-initiative dialogues [VoiceXML Forum n.d.]. VoiceXML provides an open environment with standardized dialogue scripting and speech grammar formats. Furthermore, because it is based on XML, a vast selection of editing and parsing tools is available, including both commercial and freely available open-source tools. A VoiceXML document specifies each interaction dialogue to be conducted by a VoiceXML interpreter. A VoiceXML document forms a conversational finite state machine, with some degree mixed-initiative that allows users in a limited way to input more than one value in a particular dialogue state. VoiceXML has been accepted as a standard for Web-based spoken dialogue systems and VoiceXML servers may well replace current proprietary development platforms for spoken dialogue systems. Further work will be required to integrate the more complex functionalities described in this survey into the next versions of the standard.

Bringing these points together, some of the issues that are likely to be important in spoken dialogue research in the next decade are

—more robust speech recognition, including the ability to perform well in noisy conditions, to deal with out-of-vocabulary words, and to integrate more closely with technologies for natural language processing;

—the use of prosody in spoken dialogue systems, both to provide more naturally sounding output and to assist recognition by identifying phrase boundaries as well as the functions of utterances;

—research concerned with component integration and with investigating the extent to which the language understanding and dialogue management components can compensate for deficiencies in speech recognition;

—investigations of the applicability of different technologies for particular application types, such as the costs and benefits of parsing using theoretically motivated grammars compared with robust and partial parsing and with more pragmatically driven methods such as concept spotting;

- studies of different approaches to dialogue management in relation to the requirements of an application, indicating, for example, where state-based methods are applicable and in which circumstances more complex approaches are required;
- the incorporation of more sophisticated approaches to dialogue management deriving from AI-based research;
- research into the use of stochastic and machine learning techniques;
- the development of multimodal dialogue systems;
- dialogue systems with Web integration.

It is unlikely that all of these issues will be addressed in commercial systems in the short term, although there is considerable interest in the commercial potential of voice commerce, involving the integration of spoken language and Internet technologies. In general, however, the emphasis in dialogue research is on developing more advanced systems and on testing theories of dialogue, while in the commercial environment the aim is to produce systems that will work in the real world. Here the performance of the system is measured not in terms of the evaluation measures applied to a laboratory prototype but in terms of its efficiency, effectiveness, usability and acceptability under real-world conditions. Factors that determine the successful deployment of a system include marketability, profitability, and user acceptance, for which considerable effort has to be directed toward managing user expectations in respect of the constraints of the technology and convincing users of the benefits of the technology.

In conclusion, as can be seen from this survey, there has been a dramatic increase in interest in spoken dialogue systems over the past decade, and there is every indication that this interest will continue, given that there are still many problems to be resolved and given the obvious benefits of the technology.

APPENDIX

A. SELECTED WORLD WIDE WEB ADDRESSES FOR DIALOGUE RESEARCH PROJECTS

The following is a list of Web sites for spoken dialogue technology, covering many of the major projects that could not be discussed within the scope of the survey. The links have been tested and were valid at the time of writing.

- AAAI Workshop on Miscommunication in Dialogue, August 1996
  http://www.cs.uwm.edu/faculty/mcroy/mnmPapers.html
- Center for PersonKommunikation (CPK), Aalborg, Denmark—member of the Danish Dialogue Project
  http://cpk.auc.dk/
- Computers that listen—examples of applications in commercial use
  http://www.voiceio.com/examples.htm
- CONVERSA—voice enabling technologies
  http://www.conversa.com
- CSLR Home Page (Center for Spoken Language Research, University of Colorado)
  http://cslr.colorado.edu
- CSLU Home Page (Center for Spoken Language Understanding, Oregon)
  http://cslu.cse.ogi.edu/
- Dialogues 2000 Project (BT and University of Edinburgh)
  http://www.ccir.ed.ac.uk/d2000/
- DISC—Spoken Language Dialogue Systems and Components Best practice in development and evaluation
  http://www.disc2.dk/
- EAGLES Project: Expert Advisory Group on Language Engineering Standards
  http://coral.lili.uni-bielefeld.de/gibbon/EAGLES/

- INRIA—dialogue projects at INRIA (France)
  http://www.inria.fr/Equipes/DIALOGUE-eng.html
- LIMSI: Projects on spoken language (France)
  http://www.limsi.fr/Recherche/TLP/projects.html
- Links to spoken dialogue systems (University of Hamburg)
  http://nats-www.informatik.uni/hamburg.de/j̃um/research/dialog/sys.html
- Microsoft User Interface Research—Persona Project, Conversational Interfaces
  http://www.research.microsoft.com/research/ui/
- Natural Interactive Systems Laboratory (NIS), Odense University, Denmark
  http://www.nis.sdu.dk/
- Research on Dialogue Processing, Ministry of Education, Science, Sports, and Culture, Japan
  http://winnie.kuis.kyoto-u.ac.jp/Text/taiwa/e-abst.html
- SIGDIAL—special interest group of ACL for dialogue and discourse
  http://www.sigdial.org/
- Speech Applications Project (Sun Microsystems)
  http://www.sun.com/research/speech/index.html
- Spoken Language Systems Group (MIT)
  http://sls-www.lcs.mit.edu/SLShome.html
- TOOT project on evaluation of spoken dialogue systems (AT&T Research)
  http://www.research.att.com/d̃iane/TOOT.html
- TRAINS Project Home Page (University of Rochester)
  http://www.cs.rochester.edu/research/trains/
- Verbmobil (Large project based in Germany on spoken language and dialogue)
  http://www.dfki.uni-sb.de/verbmobil/overview-us.html
- VoiceXML Forum
  http://www.voicexml.org
- Waxholm dialog project (Sweden)
  http://www.speech.kth.se/waxholm/waxholm2.html

B. WORLD WIDE WEB ADDRESSES FOR SPOKEN DIALOGUE TOOLKITS

- CPK Generic Dialogue System Platform
  http://www.kom.auc.dk/l̃bl/IMM/S9_98/SDS_course_overview.html
- CSLU toolkit
  http://cslu.cse.ogi.edu/toolkit/
- CU Communicator system
  http://cslr.colorado.edu/beginweb/cumove/cucommunicator.html
- GULAN—CCT/KTH
  http://www.speech.kth.se/j̃ocke/publications/gulan.html
- IBM Voice Server
  http://www-4.ibm.com/software/speech/
- Natural Language Speech Assistant (NLSA)(Unisys Corporation)
  http://www.unisys.com/
- NUANCE Developers Toolkit
  http://www.nuance.com/
- SpeechMania^{TM} (Philips)
  http://www.speech.be.philips.com/
- SpeechWorks
  http://www.speechworks.com/
- Vocalis SpeechWare^{TM}
  http://www.vocalis.com/

ACKNOWLEDGMENTS

I would like to acknowledge the helpful comments on an earlier draft of this paper that I received from many friends and colleagues, in particular, from Harald Aust, Alex Schoenmakers, Ronnie Smith, David James, and Ian O'Neill, and from the anonymous reviewers of the paper.

REFERENCES

Abney, S. 1997. Part-of-speech tagging and partial parsing. In *Corpus-Based Methods in Language and Speech Processing*, S. Young and G. Bloothooft, Eds. Kluwer Academic Publishers, Dordrecht, The Netherlands, 118–136.

ALLEN, J. 1983. Recognising intentions from natural language utterances. In *Computational Models of Discourse*, M. Brady and R. Berwick, Eds. MIT Press, Cambridge, MA, 107–166.

ALLEN, J. 1995. *Natural Language Processing*, 2nd ed. Benjamin Cummings Publishing Company Inc., Redwood, CA.

ALLEN, J., BYRON, D., DZIKOVSKA, M., FERGUSON, G., GALESCU, L., AND STENT, A. 2000. An architecture for a generic dialogue shell. *Natural Language Engineering 6*, 3, 1–16.

ALLEN, J., MILLER, B., RINGGER, E., AND SIKORSKI, T. 1996. A robust system for natural spoken dialogue. In *Proceedings of the 34th Annual Meeting of the ACL* (Santa Cruz, CA). ACL, 62–70.

ALLEN, J. AND PERRAULT, C. 1980. Analysing intention in utterances. *Artificial Intelligence 15*, 143–178.

ALLEN, J., SCHUBERT, L., FERGUSON, G., HWANG, C., KATO, T., LIGHT, M., MILLER, B., POESIO, M., AND TRAUM, D. 1995. The TRAINS project: a case study in building a conversational planning agent. *Journal of Experimental and Theoretical Artificial Intelligence 7*, 7–48.

ARETOULAKI, M. AND LUDWIG, B. 1999. Automaton-descriptions and theorem-proving: a marriage made in heaven? In *Proceedings of IJCAI'99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems* (Stockholm, Sweden), IJCAI.

AUST, H. AND OERDER, M. 1995. Dialogue control in automatic inquiry systems. In *Proceedings of the ESCA Workshop on Spoken Dialogue Systems*, P. Dalsgaard, L. Larsen, L. Boves, and I. Thomsen, Eds. ESCA, Vigso, Denmark, 121–124.

AUST, H., OERDER, M., SEIDE, F., AND STEINBISS, V. 1995. The Philips automatic train timetable information system. *Speech Communication 17*, 249–262.

AUSTIN, J. L. 1962. *How to Do Things with Words*. Oxford University Press, Oxford, UK.

BAGGIA, P. 1996. Evaluation of spoken dialogue systems. Turorial, The 14th European Summer School on Language and Speech Communication.

BERNSEN, N. 1993. The structure of the design space. In *Computers, Communication, and Usability: Design Issues, Research and Methods for Integrated Services*, P. Byerley, P. Barnard, and J. May, Eds. North Holland, Amsterdam, The Netherlands, 221–244.

BERNSEN, N. 1994. Foundations of multimodal representations: a taxonomy of representational modalities. *Interacting with Computers 6*, 4, 347–371.

BERNSEN, N., DYBKJÆR, H., AND DYBKJÆR, L. 1996. Co-operativity in human-machine and human-human spoken dialogue. *Discourse Processes 21*, 2, 213–236.

BERNSEN, N., DYBKJÆR, H., AND DYBKJÆR, L. 1998. *Designing Interactive Speech Systems: From First Ideas to User Testing*. Springer Verlag, New York, NY.

BILLI, R., CASTAGNERI, G., AND DANIELI, M. 1996. Field trial evaluation of two different information inquiry systems. In *IVTTA*. IEEE, Basking Ridge, NJ, 129–132.

BOROS, M., ECKERT, W., GALLWITZ, F., GÖRZ, G., HANRIEDER, G., AND NIEMANN, H. 1996. Towards understanding spontaneous speech: word accuracy vs. concept accuracy. In *Proceedings of the 4th International Conference on Spoken Language Processing* (ICSLP96), Philadephia, PA). ICSLP, 1005–1008.

BRATMAN, M., ISRAEL, D., AND POLLACK, M. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence 4*, 2, 349–355.

CARBERRY, S. 1986. The use of inferred knowledge in handling pragmatically ill-formed queries. In *Communication Failure in Dialogue*, R. Reilly, Ed. Elsevier Science Publishers North Holland, Amsterdam, The Netherlands, 187–200.

CARBERRY, S. 1989. Plan recognition and its use in understanding dialogue. In *User Models in Dialog Systems*, A. Kobsa and W. Wahlster Eds. Springer Verlag, London, UK, 133–162.

CARBERRY, S. AND LAMBERT, L. 1999. A process model for recognising communicative acts and modeling negotiation subdialogues. *Computational Linguistics 25*, 1, 1–54.

CARLETTA, J. 1996. Assessing the reliability of subjective codings. *Computational Linguistics 22*, 2, 249–254.

CARLSON, R. AND GRANSTRÖM, B. 1997. Speech synthesis. In *The Handbook of Phonetic Science*, W. J. Hardcastle and J. Laver, Eds. Blackwell, Oxford, UK, 768–788.

CHIN, D. 1989. KNOME: Modeling what the user knows in UC. In *User Models in Dialog Systems*, A. Kobsa and W. Wahlster, Eds. Springer Verlag, London, UK, 74–107.

CLARK, H. 1992. *Arenas of Language Use*. University of Chicago Press, Chicago, IL.

COHEN, P. 1994. Models of dialogue. In *Proceedings of the 4th NEC Research Symposium*, M. Nagao, Ed. SIAM Press Philadephia, PA.

COHEN, P. AND LEVESQUE, H. 1990. Rational interaction as the basis for communication. In *Intentions in Communication*, P. Cohen, J. Morgan and M. Pollack, Eds. MIT Press, Cambridge, MA, 221–256.

COHEN, P. AND OVIATT, S. 1995. The role of voice in human-machine communication. In *Voice Communication Between Humans and Machines*, D. Roe and J. Wilpon, Eds. National Academy Press, Washington, DC, 34–75.

COLE, R., MASSARO, D., DE VILLIERS, J. RUNDLE, B., SHOBAKI, K. WOUTERS, J., COHEN, M., BESKOW, J., STONE, P., CONNORS, P., TARACHOW, A., AND SOLCHER, D. 1999a. New tools for interactive speech and language training: Using animated conversational agents in the classrooms of profoundly deaf children. In *Proceedings of ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education* (London). 45–52.

COLE, R., SERRIDGE, B., HOSOM, J. P., CRONK, A., AND KAISER, E. 1999b. A platform for multilingual research in spoken dialogue systems. In *Multi-Lingual Interoperability in Speech Technology* (*MIST*) (Leusden, The Netherlands).

COLE, R., NOVICK, D., VERMEULEN, P., SUTTON, S., FANTY, M., WESSELS, L., DE VILLIERS, J., SCHALKWYK, J., HANSEN, B., AND BURNETT, D. 1997. Experiments with a spoken dialogue system for taking the U.S. census. In *Speech Communication 23*, 3, 243–260.

CONSTANTINIDES, P., HANSMA, S., TCHOU, C., AND RUDNICKY, A. 1998. A schema based approach to dialog control. In *Proceedings of the 5th International Conference on Spoken Language Processing* (ICSLP'98, Sydney, Australia), Vol. 2. ICSLP, 409–412.

DAHLBÄCK, N. AND JÖNSSON A. 1999. Knowledge sources in spoken dialogue systems. In *Proceedings of 6th European Conf. on Speech Communication and Technology* (Eurospeech'99, Budapest, Hungary). ESCA.

DALE, R. AND REITER, E. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science 19*, 233–263.

DANIELI, M. AND GERBINO, E. 1995. Metrics for evaluating dialogue strategies in a spoken language system. In *Working Notes of the AAAI Spring Symposium on Empirical Methods on Discourse Interpretation and Generation*. AAAI, Stanford, CA, 34–39.

DENECKE, M. AND WAIBEL, A. 1997. Dialogue strategies guiding users to their communicative goals. In *Proceedings of 5th European Conf. on Speech Communication and Technology* (Eurospeech'97, Rhodes, Greece). ESCA.

DOWDING, J., GANRON, J. M., APPELT, D., BEAR, J. CHERNY, L., MOORE, R., AND MORAN, D. 1993. Gemini: a natural language system for spoken language understanding. In *Proceedings of the 31st Annual Meeting of the ACL*. ACL Columbus, OH, 54–61.

DYBKJÆR, L., BERNSEN, N. O., AND DYBKJÆR, H. 1996. Evaluation of spoken dialogue systems. In *Proceedings of the Eleventh Twente Workshop on Language Technology (TWLT 11): Dialogue Management in Natural Language Systems*, S. LuperFoy and A. Nijholt and G. V. van Zanten, Eds. Universiteit Twente, Enschede, The Netherlands.

DYBKJÆR, L., BERNSEN, N. O., AND DYBKJÆR, H. 1997. Generality and objectivity: central issues in putting a dialogue evaluation tool into practical use. In *Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications. Proceedings of a Workshop Sponsored by the Association for Computational Linguistics* (Madrid, Spain), J. Hirschberg, C. Kamm and M. Walker, Eds. ACL, 17–24.

DYBKJÆR, L. BERNSEN. N. O., AND DYBKJÆR, H. 1998. A methodology for diagnostic evaluation of spoken human-machine interaction. *International Journal of Human-Computer Studies 48*, 605–625.

ECKERT, W. AND NIEMANN, H. 1994. Semantic analysis in a robust spoken dialog system. In *Proceedings of the 3rd International Conference on Spoken Language Processing* (Yokohama, Japan). ICSLP, 107–110.

ECKERT, W., NÖTH, E., NIEMANN, H., AND SCHUKAT-TALAMAZZANI, E. G. 1995. Real users behave weird experiences made collecting large human-machine dialog corpora. In *Proceedings of the ESCA Workshop on Spoken Dialogue Systems*, P. Dalsgaard, L. Larsen, L. Boves, and I. Thomsen, Eds. ESCA, Vigso, Denmark, 193–196.

EDGINGTON, M., LOWRY, A., JACKSON, P., BREEN, A. P., AND MINNIS, S. 1996a. Overview of current text-to-speech synthesis techniques: Part I—text and linguistic analysis. *BT Technology Journal 14*, 1, 68–83.

EDGINGTON, M., LOWRY, A., JACKSON, P., BREEN, A. P., AND MINNIS, S. 1996b. Overview of current text-to-speech synthesis techniques: Part II—prosody and speech generation. *BT Technology Journal 14*, 1, 84–99.

FERGUSON, G., ALLEN, J. F., AND MILLER, B. 1996. Trains-95: Towards a mixed-initiative planning assistant. *Proceedings of the 3rd International Conference on AI Planning Systems* (AIPS-96, Edinburgh, Scotland, UK). 70–77.

FRASER N. 1997. Assessment of interactive systems. In *Handbook of Standards and Resources for Spoken Language Systems*, D. Gibbon, R. Moore, and R. Winski, Eds. Mouton de Gruyter, New York, NY, 564–614.

FRASER, N. AND GILBERT, G. N. 1991. Simulating speech systems. *Computer Speech and Language 5* 81–99.

GERBINO, E. AND DANIELI, M. 1993. Managing dialogue in a continuous speech understanding system. In *Proceedings of 3rd European Conference on Speech Communication and Technology* (Eurospeech'93, Berlin, Germany). ESCA, 1661–1664.

GIACHIN, E. AND MCGLASHAN, S. 1997. Spoken language dialogue systems. In *Corpus-Based Methods in Language and Speech Processing*, S. Young and G. Bloothooft, Eds. Kluwer Academic Publishers, Dordrecht, The Netherlands, 69–117.

GIBBON, D. MOORE, R., AND WINSKI R., EDS. 1997. *Handbook of Standards and Resources for Spoken Language Systems*. Mouton de Gruyter, New York, NY.

GODDEAU, D., MENG, H., POLIFRONI, J., SENEFF, S., AND BUSAYAPONGCHAI. 1996. A form-based dialogue manager for spoken language applications. In *Proceedings of 4th International Conference on Spoken Language Processing* (ICSLP'96, Pittsburgh, PA). ICSLP, 701–704.

GRICE, P. 1975. Logic and conversation. In *Syntax and Semantics Vol. 3: Speech Acts*, P. Cole and J. Morgan, Eds. Academic Press, New York, NY, 41–58.

GROSZ, B. J., JOSHI, A. K., AND WEINSTEIN, S. 1983. Providing a unified account of definite noun phrases in discourse. In *Proceedings of the 21st Annual Meeting of the ACL*. ACL, Boston, MA, 44–50.

GROSZ, B. J. AND SIDNER, C. 1986. Attention, intention, and the structure of discourse. In *Computational Linguistics 12*, 3, 175–204.

HANSEN, B., NOVICK, D. G., AND SUTTON, S. 1996. Systematic design of spoken prompts. In *CHI'96* (Vancouver, B.C., Canada). ACM Press, New York, NY, 157–164.

HEEMAN, P. A. AND ALLEN, J. F. 1997. Intonational boundaries, speech repairs, and discourse markers: modeling spoken dialog. In *Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the Association for Computational Linguistics* (Madrid, Spain). ACL, 254–261.

HEISTERKAMP, P. AND MCGLASHAN, S. 1996. Units of dialogue management: an example. In *Proceedings of the 4th International Conference on Spoken Language Processing* (ICSLP'96, Philadelphia, PA). ICSLP, 200–203.

HIRSCHMAN, L. 1995. The roles of language processing in a spoken language interface. In *Voice Communication Between Humans and Machines*, D. Roe and J. Wilpon, Eds. National Academy Press Washington, DC, 217–237.

HONE, K. S. AND BABER, C. 1995. Using a simulation method to predict the transaction time effects of applying alternative levels of constraint to utterances within speech interactive dialogues. In *Proceedings of the ESCA Workshop on Spoken Dialogue Systems*, P. Dalsgaard, L. Larsen, L. Boves, and I. Thomsen, Eds. ESCA, Vigso, Denmark, 209–212.

JAMESON, A. 1989. But what will the user think? Belief ascription and image maintenance in dialog. In *User Models in Dialog Systems*, A. Kobsa and W. Wahlster, Eds. Springer Verlag, London, UK, 255–312.

JURAFSKY, D. AND MARTIN, J. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing*. Prentice Hall, Englewood Cliffs, NJ.

KAISER, E. C., JOHNSTON, M., AND HEEMAN, P. A. 1999. Profer: Predictive, robust finite-state parsing for spoken language. In *Proceedings of ICASSP* (Phoenix, AZ), Vol. 2. IEEE, 629–632.

KAMM, C. 1995. User interfaces for voice applications. In *Voice Communication Between Humans and Machines*, D. Roe and J. Wilpon, Eds. National Academy Press, Washington, DC, 34–75.

KAMM, C., WALKER, M. A., AND J. LITMAN, D. 1999. Evaluating spoken language systems. In *Proceedings of American Voice Input/Output Society (AVIOS)*. AVIOS.

KAPLAN, J. 1983. Cooperative responses from a portable natural language database query system. In *Computational Models of Discourse*, M. Brady and R. Berwick, Eds. MIT Press, Cambridge, MA, 167–208.

KUBALA, F., BARRY, C., BATES, M., BOBROW, R., FUNG, P., INGRIA, R., MAKHOUL, J., NGUYEN, L., SCHWARTZ, R., AND STALLARD, D. 1992. BBN Byblos and HARC February 1992 ATIS benchmark results. In *Proceedings of the DARPA Speech and Natural Language Workshop, Harriman, N.Y.* Morgan Kaufmann Publishers, San Mateo, CA, 72–77.

LAMEL, L. F., BENNACEF, S. K., BONNEAU-MAYNARD, H., ROSSETN, S., AND GAUVAIN, J. L. 1995. Recent developments in spoken language systems for information retrieval. In *Proceedings of the ESCA Workshop on Spoken Dialogue Systems*, P. Dalsgaard, L. Larsen, L. Boves, and I. Thomsen, Eds. ESCA, Vigso, Denmark, 17–20.

LARSEN, L. B., AND BAEEKGAARD, A. 1994. Rapid prototyping of a dialogue system using a generic dialogue development platform. In *Proceedings of ICSLP'94* (Yokohama, Japan). ICSLP, 919–922.

LENNIG, M., BIELBY, G., AND MASSICOTTE, J. 1995. Directory assistance automation in Bell Canada: Trial results. *Speech Communication 17*, 227–234.

LITMAN, D. J. AND ALLEN, J. F. 1987. A plan recognition model for subdialogues in conversation. *Cognitive Science 11*, 163–200.

LITMAN, D. J., KEARNS, M. S., SINGH, S., AND WALKER, M. A. 2000. Automatic optimization of dialogue management. In *Proceedings of 18th International Conference on Computational Linguistics* (COLING-2000, Saarbrucken, Germany). ACL.

MAKHOUL, J. AND SCHWARTZ, R. 1995. State of the art in continuous speech recognition. In *Voice Communication Between Humans and Machines*, D. Roe and J. Wilpon, Eds. National Academy Press, Washington, DC, 165–197.

MANN, W. AND THOMPSON, S. 1988. Rhetorical structure theory: toward a functional theory of text organisation. *Text 3*, 243–281.

MARCUS, M. 1995. New trends in natural language processing: statistical natural language processing. In *Voice Communication Between Humans and Machines*, D. Roe and J. Wilpon, Eds. National Academy Press, Washington, DC, 482–504.

MARTIN, P., CRABBE, F., ADAMS, S., BAATZ, E., AND YANKELOVICH, N. 1996. SpeechActs: a spoken language framework. *IEEE Computer 29*, *7*, 33–40.

MAYBURY, M. T., Ed. 1993. *Intelligent Multimedia Interfaces*. MIT Press, Cambridge, MA.

McCoy, K. F. 1986. Generating responses to property misconceptions using perspective. In *Communication Failure in Dialogue*, R. Reilly, Ed. Elsevier Science Publishers North Holland, Amsterdam, The Netherlands, 149–160.

McGlashan, S., Bilange, E., Fraser, N., Gilbert, N., Heisterkamp, P., and Youd, N. 1990. Managing oral dialogues. Research Report, Social and Computer Sciences Research Group, University of Surrey, Surrey, UK.

McKeown, K. 1985. *Text Generation*. Cambridge University Press, Cambridge, UK.

McRoy, S. 1996. Detecting, repairing, and preventing human-machine miscommunication. In *AAAI '96 Workshop* (Portland, OR).

McTear, M. 1998. Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. In *Proceedings of the 5th International Conference on Spoken Language Processing* (ICSLP'98, Sydney, Australia). ICSLP, 1223–1226.

McTear, M. 1999. Using the CSLU toolkit for practicals in spoken dialogue technology. In *Proceedings of ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education* (London, UK). ESCA, 113–116.

McTear, M., Allen, S., Clatworthy, L., Ellison, N., Lavelle, C., and McCaffery, H. 2000. Integrating flexibility into a structured dialogue model: Some design considerations. In *Proceedings of the 6th International Conference on Spoken Language Processing* (ICSLP'2000, Beijing, China), Vol. 1, ICSLP, 110–113.

Moore, R. C. 1995. Integration of speech with natural language understanding. In *Voice Communication Between Humans and Machines*, D. Roe and J. Wilpon, Eds. National Academy Press, Washington, DC, 254–271.

Nagata, M. and Morimoto, T. 1994. First steps toward statistical modeling of dialogue to predict the speech act type of the next utterance. *Speech Communication 15*, 193–203.

Page, J. H. and Breen, A. P. 1996. The Laureate text-to-speech system—architecture and applications. *BT Technology Journal 14*, 1, 57–67.

Paris, C. L. 1989. The use of explicit user models in a generation system for tailoring answers to the user's level of expertise. In *User Models in Dialog Systems*, A. Kobsa and W. Wahlster, Eds. Springer Verlag, London, UK, 200–232.

Peckham, J. 1993. A new generation of spoken dialogue systems: Results and lessons from the SUNDIAL project. In *Proceedings of 3rd European Conference on Speech Communication and Technology* (Eurospeech'93, Berlin, Germany). ESCA, 33–40.

Peckham, J. n.d. Vocalis develops directory enquiries service with speech recognition for Telia. `http://www.callcentres.com.au/speechr1.htm`.

Philips Speech Processing. 1997. Hddl v2.0—dialog description language—user's guide. Philips Speech Processing, Aachen, Germany.

Pollack, M. 1986. Some requirements for a model of the plan-inference process in conversation. In *Communication Failure in Dialogue*, R. Reilly, Ed. Elsevier Science Publishers North Holland, Amsterdam, The Netherlands, 245–256.

Potjer, J., Russel, A., Boves, L., and Os, E. D. 1996. Subjective and objective evaluation of two types of dialogues in a call assistance service. In *IVTTA*. IEEE, Basking Ridge, NJ, 121–124.

Power, K. J. 1996. The listening telephone—automating speech recognition over the PSTN. *BT Technology Journal 14*, 1, 112–126.

Price, P. 1996. Spoken language understanding. In *Survey of the State of the Art in Human Language Technology*, R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, Eds. Cambridge University Press, Cambridge, UK. Online version: `http://cslu.cse.ogi.edu/HLTsurvey/`.

Pulman, S. G. 1996. Semantics. In *Survey of the State of the Art in Human Language Technology*, R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, Eds. Cambridge University Press, Cambridge, UK. Online version: `http://cslu.cse.ogi.edu/HLTsurvey/`.

Rabiner, L. R. and Juang, B. H. 1993. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ.

Reiter, E. and Dale, R. 1997. Building applied natural language generation systems. *Natural Language Engineering 3*, 1, 57–87.

Rudnicky, A. I., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu, W., and Oh, A. 1999. Creating natural dialogs in the Carnegie Mellon Communicator system. In *Proceedings of 6th European Conference on Speech Communication and Technology* (Eurospeech'99, Budapest, Hungary). ESCA.

Sadek, M. D., Bretier, P., and Panaget, F. 1997. ARTIMIS: Natural dialogue meets rational agency. In *Proceedings of 15th International Joint Conference on Artificial Intelligence,* (*IJCAI-97*). Morgan Kaufmann Publishers, San Francisco, CA, 1030–1035.

Sadek, M. D. and de Mori, R. 1998. Dialogue systems. In *Spoken Dialogues with Computers*, R. de Mori, Ed. Academic Press, London, UK, 523–561.

Searle, J. R. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, UK.

Seneff, S. 1992. TINA: A natural language system for spoken language applications. *Computational Linguistics 18*, 1, 61–86.

Shieber, S. M. 1986. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes, CSLI, Stanford, CA.

SIMPSON, A. AND FRASER, N. 1993. Black box and glass box evaluation of the SUNDIAL system. In *Proceedings of 3rd European Conference on Speech Communication and Technology* (Eurospeech'93, Berlin, Germany). ESCA, 33–40.

SJÖLANDER, K., BESKOW, J., GUSTAFSON, J., LEWIN, E., CARLSON, R., AND GRANSTRÖM, B. 1998. Web-based educational tools for speech technology. In *Proceedings of the 5th International Conference on Spoken Language Processing* (ICSLP'98, Sydney, Australia). ICSLP, 3217–3220.

SMITH, R. AND GORDON, S. A. 1997. Effects of variable initiative on linguistic behavior in human-computer spoken natural language dialogue. *Computational Linguistics 23*, 1, 141–168.

SMITH R. AND HIPP D. R. 1994 *Spoken Natural Language Dialog Systems: A Practical Approach*. Oxford University Press, New York, NY.

SMITH, R. W. 1997. Performance measures for the next generation of spoken natural language dialog systems. In *Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications. Proceedings of a Workshop Sponsored by the Association for Computational Linguistics* (Madrid, Spain), J. Hirschberg, C. Kamm, and M. Walker, Eds. ACL, 37–40.

STALLARD, D. AND BOBROW, R. 1992. Fragment processing in the DELPHI system. In *Proceedings of the Speech and Natural Language Workshop, Harriman, N.Y.* Morgan Kaufmann Publishers, San Mateo, CA, 305–310.

STRIK, H., RUSSEL, A., VAN DEN HEUVEL, H., CUCCHIARINI, C., AND BOVES, L. 1996. Localizing an automatic inquiry system for public transport information. In *Proceedings of the 4th International Conference on Spoken Language Processing* (ICSLP'96, Philadephia, PA), Vol. 2, ICSLP, 24–31.

SUTTON, S., COLE, R., DE VILLIERS, J., SCHALKWYK, J., VERMEULEN, P., MACON, M., YAN, Y., KAISER, E., RUNDLE, B., SHOBAKI, K., HOSOM, J. P., KAIN, A., WOUTERS, J., MASSARO, D., AND COHEN, M. 1998. Universal speech tools: The CSLU toolkit. In *Proceedings of the 5th International Conference on Spoken Language Processing* (ICSLP'98, Sydney, Australia). ICSLP, 3221–3224.

SUTTON, S., HANSEN, B., LANDER, T., NOVICK, D. G., AND COLE, R. 1995. Evaluating the effectiveness of dialogue for an automated spoken questionnaire. Tech. Rep. CS/E95-12, Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology.

TRAUM, D. R. 1996. Conversational agency: the TRAINS-93 dialogue manager. In *Proceedings of the eleventh Twente Workshop on Language Technology (TWLT 11): Dialogue Management in Natural Language Systems*, S. LuperFoy, A. Nijholt, and G. V. van Zanten, Eds. Universiteit Twente, Enschede, The Netherlands.

TRAUM, D. R. AND ALLEN, J. F. 1994. Discourse obligations in dialogue processing. In *Proceedings of the 32nd Annual General Meeting of the Association for Computational Linguistics* (Las Cruces, NM). ACL, 1–8.

USZKOREIT, H. AND ZAENEN, A. 1996. Grammar formalisms. In *Survey of the State of the Art in Human Language Technology*, R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, Eds. Cambridge University Press, Cambridge, UK. Online version: `http://cslu.cse.ogi.edu/HLTsurvey/`.

VERGEYNST, N. A., EDWARDS, K., FOSTER, J. C., AND JACK, M. A. 1993. Spoken dialogues for human-computer interaction over the telephone: complexity measures. In *Proceedings of 3rd European Conference on Speech Communication and Technology* (Eurospeech'93, Berlin, Germany). ESCA, 1415–1418.

VOICEXML FORUM. n.d. `http://www.voicexml.org`.

WAHLSTER, W. AND KOBSA, A. 1989. User models in dialog systems. *User Models in Dialog Systems*, A. Kobsa and W. Wahlster, Eds. Springer Verlag, London, UK, 4–24.

WALKER, M. A. 1989. Evaluating discourse processing algorithms. In *Proceedings of the 27th Annual General Meeting of the Association for Computational Linguistics* (Vancouver, B.C., Canada). ACL, 251–261.

WALKER, M. A., LITMAN, D., KAMM, C., AND ABELLA, A. 1997. PARADISE: a general framework for evaluating spoken dialogue agents. In *Proceedings of the 35th Annual General Meeting of the Association for Computational Linguistics, ACL/EACL* (Madrid, Spain). ACL, 271–280.

WALKER, M. A., LITMAN, D., KAMM, C., AND ABELLA, A. 1998. Evaluating spoken dialogue agents with PARADISE: Two case studies. *Computer Speech and Language 12*, 3, 317–347.

WARD, W. AND PELLOM, B. 1999. The CU Communicator system. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding* IEEE, (Keystone, CO). IEEE.

WHITTAKER, S. J. AND ATTWATER, D. J. 1996. The design of complex telephony applications using large vocabulary speech technology. In *Proceedings of the 4th International Conference on Spoken Language Processing* (ICSLP-96, Philadephia, PA). ICSLP, 705–708.

WRIGHT, J., GORIN, A., AND ABELLA, A. 1998. Spoken language understanding within dialogs using a graphical model of task structure. In *Proceedings of the 5th International Conference on Spoken Language Processing* (ICSLP'98, Sydney, Australia), Vol. 5. ECSLP.

WYARD, P. J., SIMONS, A. D., APPELBY, S., KANEEN, E., WILLIAMS, S. H., AND PRESTON, K. R. 1996. Spoken language systems—beyond prompt and response. *BT Technology Journal 14*, 1, 187–205.

YANKELOVICH, N. n.d. Using natural dialogues as the basis for speech interface design. In *Automated Spoken Dialog Systems*, S. Luperfoy, Ed. MIT Press, Cambridge, MA.

YANKELOVICH, N., LEVOW, G., AND MARX, M. 1995. Designing SpeechActs: issues in speech user interfaces. In *Proceedings of CHI95*. Addison-Wesley, Reading, MA. 369–375.

YOUNG, S. AND BLOOTHOOFT, G. EDS. 1997. *Corpus-Based Methods in Language and Speech Processing*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

YOUNG, S. R., HAUPTMANN, A. G., WARD, W. H., SMITH, E. T., AND WERNER, P. 1989. High level knowledge sources in usable speech recognition systems. *Communications of the ACM 32*, 2, 183–194.

Received September 1998; revised February 2000, October 2000; accepted October 2001