19 DIALOGUE AND CONVERSATIONAL AGENTS

C: I want you to tell me the names of the fellows on the St. Louis team.
A: I'm telling you. Who's on first, What's on second, I Don't Know is on third.
C: You know the fellows' names?
A: Yes.
C: Well, then, who's playing first?
A: Yes.
C: I mean the fellow's name on first.
A: Who.
C: The guy on first base.
A: Who is on first.
C: Well what are you askin' *me* for?
A: I'm not asking you – I'm telling you. Who is on first.

Who's on First – Bud Abbott and Lou Costello's version of an old burlesque standard.

The literature of the fantastic abounds in inanimate objects magically endowed with sentience and the gift of speech. From Ovid's statue of Pygmalion to Mary Shelley's Frankenstein, Cao Xue Qin's Divine Luminescent Stone-in-Waiting in the Court of Sunset Glow to Snow White's mirror, there is something deeply touching about creating something and then having a chat with it. Legend has it that after finishing his sculpture of *Moses*, Michelangelo thought it so lifelike that he tapped it on the knee and commanded it to speak. Perhaps this shouldn't be surprising. Language itself has always been the mark of humanity and sentience, and **conversation** or **dialogue** is the most fundamental and specially privileged arena of language. It is certainly the first kind of language we learn as children, and for most of us, it is the kind of language we most commonly indulge in, whether we are ordering curry for lunch or buying postage stamps, participating in business meetings or talking with our families, booking airline flights or complaining about

CONVERSATION
DIALOGUE

the weather.

This chapter introduces the fundamental structures and algorithms in **conversational agents**, also known as **spoken dialogue systems**, or **spoken language systems**. These are programs which communicate with users in spoken natural language in order to make travel arrangements, answer questions about weather or sports, route telephone calls, act as a general telephone assistant, or perform even more sophisticated tasks.

Many of these tasks have to do with telephony; speech dialogue systems are particularly appropriate for applications like telephony, where only a limited keyboard and/or screen is available (Cohen and Oviatt, 1995). One important domain for conversational agents is travel planning, which has been a key concern of dialogue systems since the early influential GUS system for planning airline travel (Bobrow et al., 1977). More recently, a number of spoken language systems for booking airline flights were built under the Air Traffic Information System (ATIS) (Hemphill et al., 1990) and Communicator (?) programs. We will use this travel domain as an example throughout the chapter (as we used it to discuss grammar rules for parsing ATIS sentences in Chapter 9). Figure 19.1 shows a sample dialogue from a Communicator system, taken from Xu and Rudnicky (2000).

S1:	Hello. You've reached the [Communicator]. Tell me your full name
U2:	*hi i'd like to fly to seattle tuesday morning*
S3:	Traveling to Seattle on Tuesday, August eleventh in the morning. Your full name?
U4:	*[John Doe]*
S5:	Hello, [John], I'm glad to hear from you again. You can now ask me for help.
S6:	OK. Let's see I have a United flight departing Pittsburgh at ten forty five a.m. arriving into Seattle at one fifty seven p.m. Is that OK?
U7:	*Yes*
S8:	Will you return to Pittsburgh from Seattle?
U9:	*no i will go to Los Angeles*
S10:	Traveling to Los Angeles. on what date
U11:	*Sorry can we go back to talk about the previous leg again*
S12:	For the first leg, you are on a United flight 1446 departing Pittsburgh at ten forty five A.M. tomorrow, arriving in Seattle at one fifty seven P.M. Do you want to take that?
U13:	*actually i want to leave on wednesday*
S14:	a flight on wednesday.
S15:	I have a flight departing Pittsburgh at ten forty five a.m., arrives Seattle at one fifty seven p.m. Is that OK?
U16:	*Yes*

Figure 19.1 The travel domain: a fragment from a successful conversation between a user (U) and the Communicator system (S) of Xu and Rudnicky (2000).

Another useful domain is automatic call routing. The task of a call routing system is to direct incoming calls in a telephone call center. The job of the system is thus to transfer the call to the appropriate human for solving the user's problem, generally beginning with open-ended questions to the user like *How may I help you?* (Gorin et al., 1997), or *How may I direct your call?* (Chu-Carroll and Carpenter, 1999). Figure 19.2 shows sample conversations from the AT&T How May I Help You (HMIHY) system (Gorin et al., 1997).

S: How may I help you? U: *can you tell me how much it is to tokyo?* S: You want to know the cost of a call? U: *yes that's right* S: Please hold on for rate information.
S: How may I help you? U: *what's the area code for chicago?* S: You want an area code. U: *(silence)* S: The area code for Chicago is 312.

Figure 19.2 The call routing/telephone information domain: two actual conversations between a user (U) and the AT&T How May I Help You (HMIHY) system (S) (Gorin et al., 1997).

A somewhat different domain, which is very far from the telephony area, is automatic tutoring systems. Figure 19.3 shows part of a dialogue from the ITSPOKE dialogue system (Litman and Silliman, 2004), which uses the Why2-Atlas text-based tutoring system as its tutoring backend(?).

Section 19.1 begins our discussion of conversational agents by overviewing the architecture of simple spoken language systems, especially frame-based dialogue systems, introducing in detail each of the components of a typical dialogue system. Section 19.3 discusses how to evaluate dialogue systems. Section 19.2 then describes the VXML (Voice XML) language for describing simple dialogue systems, and walks through some examples using form-based dialogues. We then turn to two sections describing more advanced concepts in dialogue. Section 19.4 introduces the idea of grounding, and the use of grounding information in dealing with confirmation and errors. Section 19.5 and Section 19.6 introduce the idea of speech acts and dialogue acts and the automatic detection of them. Finally we discuss some advanced topics, including the advanced BDI (belief-desire-intention) paradigm for dialogue understanding, the MDP (Markov Decision Process) model for dialogue understanding, and some discussions of speech processing of human-human dialogue, for example for automatic summarization of meetings.

S: I didn't understand. What force acts on all objects within close proximity to earth?
U: *uh the force of gravity*
S: Fine. Besides the Earth's gravitational force, are there any other forces acting on the person, keys, and elevator after the release?
U: *no*
S: Yep. Thus, keep in mind that after the release, the only force on the person, keys, and elevator is the force of gravity. If gravity is the only force acting on an object, will it be moving or staying still?
U: *it will be moving*
S: Fine. Not only are the person, keys, and elevator moving, they have only gravitational forces on them. When an object is falling and has only gravitational force on it, it is said to be in what?
U: *freefall*

Figure 19.3 The tutoring domain: part of an actual conversation between a student user (U) and the ITSPOKE system (S) of (Litman and Silliman, 2004), based on the Why2-Atlas text-based tutoring system (?).

19.1 SIMPLE DIALOGUE SYSTEMS

The very simplest dialogue system, albeit one for text rather than speech, is one which we explored already in Chapter 2 for chatterbots like Eliza. The Eliza architecture consisted of a single read-search-replace-print loop, which read in a user sentence, ran a series of regular expression substitutions, and printed out the resulting response. The dialogue systems we are concerned with in this chapter are vastly more complex, with speech input and output and more sophisticated dialogue control. Figure 19.4 shows a typical architecture.

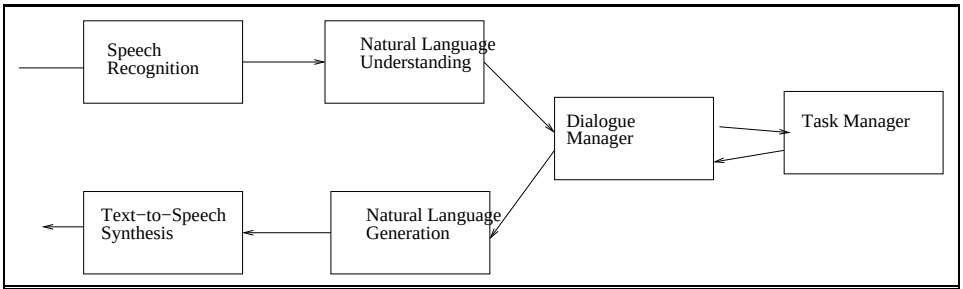

Figure 19.4 Simplified architecture of the components of a conversational agent. PLACEHOLDER FIGURE.

ASR component

The ASR (automatic speech recognition) component takes audio input, generally from the telephone, and returns a transcribed string of words, as discussed in chapters Chapter 6 through Chapter 7. The ASR system may also be optimized in various ways for use in conversational agents. For example while ASR systems used for dictation or transcription generally use a single broadly-trained N-gram language model, ASR systems in conversational agent generally use more specific language models. These language models are specific to a dialogue state. For example, if the system has just asked the user "What city are you departing from?", the ASR language model can be constrained to only consist of city names, or perhaps sentences of the form 'I want to (leave|depart) from [CITYNAME]'. These dialogue-state-specific language models can consist of hand-written finite-state or context-free grammars, or of N-gram grammars trained on subcorpora extracted from the answers to particular questions in some training set. When the system wants to constrain the user to respond to the system's last utterance, it can use such a **restrictive grammar**. When the system wants to allow the user more options, it might mix this state-specific language model with a more general language model. As we will see, the choice between these strategies can be tuned based on how much *initiative* the user is allowed.

RESTRICTIVE GRAMMAR

Another way that ASR is influenced by being embedded in a dialogue system has to do with adaptation. Since the identity of the user remains constant across the telephone call, speaker adaptation techniques can be applied to improve recognition as the system hears more and more speech from the user. Thus techniques like MLLR and VTLN can provide useful improvements in ASR rates in a dialogue situation.

NLU component

The NLU (natural language understanding) component of dialogue systems must produce a semantic representation which is appropriate for the dialogue task. Many speech-based dialogue systems, since as far back as the GUS system (Bobrow et al., 1977), have used frames as their semantic representation, exactly the kind of frame-and-slot semantics that are commonly used in information extraction, as discussed in Chapter 15. A travel system, for example, which has the goal of helping a user find an appropriate flight, would have a frame with slots for information about the flight; thus a sentence like *Show me morning flights from Boston to San Francisco on Tuesday* might correspond to the following filled-out frame (from Miller et al. (1994)):

```
SHOW:
```

```
FLIGHTS:
    ORIGIN:
        CITY:  Boston
        DATE:
            DAY-OF-WEEK:  Tuesday
        TIME:
            PART-OF-DAY:  morning
    DEST:
        CITY:  San Francisco
```

How does the NLU component generate this semantics? In principle any of the methods for semantic analysis discussed in Chapter 15 could be employed. For example, Chapter 15 shows how a context-free grammar could be augmented with semantic attachments, and a standard CFG parser can be used to build a meaning for a sentence. The fillers of each frame can then be extracted from the sentence meaning. For example the SRI GEMINI NLU engine, a unification grammar with semantic attachments, is used in the ATIS and WITAS dialogue systems (?, ?)

In practice, most dialogue systems rely on a simpler, domain-specific semantic understanding component. Probably the most common is the use of the semantic grammars, also discussed in Chapter 15. With these grammars, rather than using a syntactic rule with a semantic attachment, the actual node names in the parse tree correspond to the semantic entities which are being expressed. For example, we might see grammar fragments like the following:

LIST	→	show me | i want | can i see|...
DEPART_TIME_RANGE	→	(after|around|before) HOUR | morning | afternoon | evening
HOUR	→	one|two|three|four...|twelve (AMPM)
FLIGHTS	→	(a) flight | flights
AMPM	→	am | pm
ORIGIN	→	from CITY
DESTINATION	→	to CITY
CITY	→	Boston | San Francisco | Denver | Washington

These grammars take the form of context-free grammars, and hence can be parsed by any standard CFG parsing algorithm, such as the CYK or Earley algorithms introduced in Chapter 10.

But grammars for these domain-specific dialogue systems are often simple enough to have minimal or no recursion, and so are often processed by efficient finite-state methods. In cases where there is some recursion, efficient augmentations of finite-state algorithms such as the use of recursive transition networks have been applied (Issar and Ward, 1993; Ward and Issar, 1994).

The result of the parse is thus a hierarchical labeling of the input string with semantic node labels:

```
                      DEPART     DESTINATION
LIST     FLIGHTS      CITY       CITY           DEPART_DATE  DEPART_TIME
Show me  flights  from boston  to san francisco  on tuesday   morning
```

Since the semantic nodes in the grammar like DEPART CITY correspond to the slots in the domain-specific frame, the fillers of each slot in the frame can be read almost directly off the resulting parse above. It remains only to put the fillers into some sort of canonical form (for example dates can be converted into a DD:MM:YY form, times can be put into 24-hour time, etc).

The semantic grammar approach is very widely used, but has two weaknesses: first, it relies on hand-written grammars, which are expensive and time-consuming to produce. Second, as we have described it so far, the semantic grammar approach is non-probabilistic, which makes it hard to resolve ambiguities.

One solution to both of these problem is to use a probabilistic model like an HMM to assign the semantic slot roles to words in the sentences. In the semantic HMM model of Pieraccini et al. (1991), for example, the hidden states of the HMM are semantic slot labels, while the observed words are the fillers of the slots. Figure 19.5 shows a sequence of hidden states, corresponding to slot names, each generating a sequence of observed words. Note that the model includes a hidden state called DUMMY which is used to generate words which do not fill any slots in the frame.

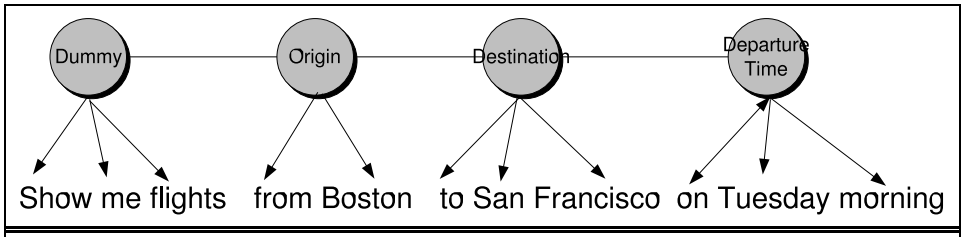

Figure 19.5 The Pieraccini et al. (1991) HMM model of semantics for filling slots in frame-based dialogue systems.

The goal of the HMM model is to compute the labeling of semantic roles $C = c_1, c_2, ..., c_i$ (C for 'cases' or 'concepts') that has the highest probability $P(C|W)$ given some words $W = w_1, w_2, ..., w_n$. As usual, we use Bayes Rules as follows:

$$\begin{aligned}
\operatorname*{argmax}_C P(C|W) &= \operatorname*{argmax}_C \frac{P(W|C)P(C)}{P(W} \\
&= \operatorname*{argmax}_C P(W|C)P(C) \qquad (19.1) \\
&= \prod_{i=2}^{N} P(w_i|w_{i-1}...w_1, C)P(w_1|C)\prod_{i=2}^{M} P(c_i|c_{i-1}...c_1) \qquad (19.2)
\end{aligned}$$

The Pieraccini et al. (1991) model makes a simplification that the concepts (the hidden states) are generated by a Markov process (a concept M-gram model), and that the observation probabilities for each state are generated by a state-dependent (concept-dependent) word N-gram word model:

$$P(w_i|w_{i-1},...,w_1,C) = P(w_i|w_{i-1},...,w_{i-N+1},c_i) \quad (19.3)$$

$$P(c_i|c_{i-1},...,c_1 = P(c_i|c_{i-1},...,c_{i-M+1}) \quad (19.4)$$

Based on this simplifying assumption, the final equations used in the HMM model are as follows:

$$\underset{C}{\operatorname{argmax}} P(C|W) = \prod_{i=2}^{N} P(w_i|w_{i-1}...w_{i-N+1},c_i) \prod_{i=2}^{M} P(c_i|c_{i-1}...c_{i-M+1}) \quad (19.5)$$

These probabilities can be trained on a labeled training corpus, in which each sentence is hand-labeled with the concepts/slot-names associated with each string of words. The best sequence of concepts for a sentence, and the alignment of concepts to word sequences, can be computed by the standard Viterbi decoding algorithm.

In summary, the resulting HMM model is a generative model with two components. One, corresponding to $P(C)$, represents the choice of what meaning to express; it assigns a prior over sequences of semantic slots, computed by a concept N-gram. The second, corresponding to $P(W|C)$, represents the choice of what words to use to express that meaning; the likelihood of a particular string of words being generated from a given slot. It is computed by a word N-gram conditioned on the semantic slot. This model is very similar to the HMM model for **named entity** detection we saw in Chapter 15.

A problem with the HMM model so far is that it has no ability to model the hierarchical nature of language structure. Various more sophisticated versions of the HMM model address this problem by augmenting the HMM with recursive structure. For example the Hidden Understanding Model (HUM) (Miller et al., 1994, 1996, 2000), is based on stochastic recursive transition networks (SRTNs), allowing the semantic labels to have hierarchy and recursion. Recall that a recursive transition network is a notational variant of a context-free grammar. Figure 19.6 shows a representation of the HUM structure of the sentence *'Please show me morning flights from Boston to San Francisco on Tuesday'*.

The model for $P(W|C)$ in the HUM model is exactly the same as in the HMM model described above: a concept-specific word N-gram model. The model for $P(C)$ is different; instead of using an N-gram model of concepts, the HUM allows for hierarchical structure by using an SCFG-like model of concept probabilities,

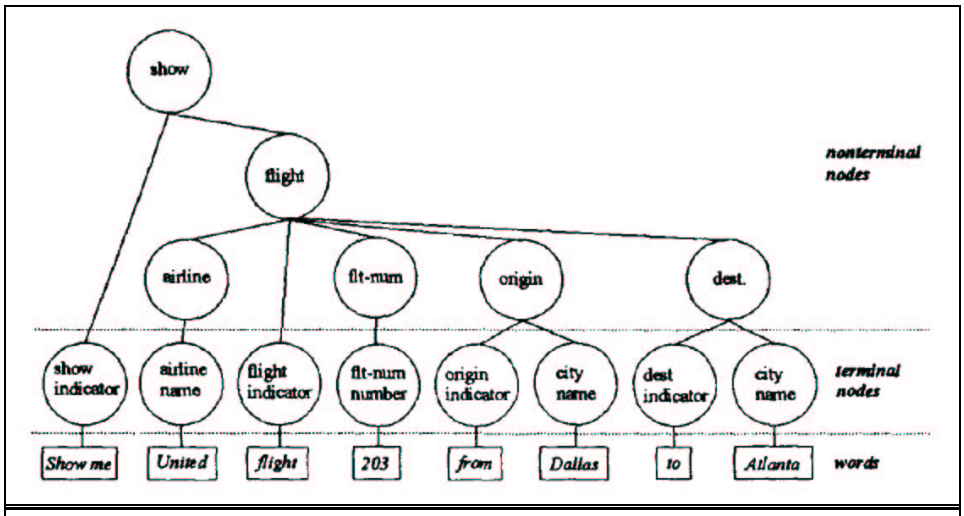

Figure 19.6 A structure of a sentence in the Hidden Understanding Model. (from (Miller et al., 1994)) PLACEHOLDER FIGURE.

borrowed from the TINA model we described below (Seneff, 1995). The probability of the concept *flight* generating the subconcept sequence *airline, flight indicator, flt-num, origin, destination* is computing by keeping a concept N-gram for the subconcept sequences, conditioned on the parents. This model is thus a hierarchical version of the model for generating word observations. Thus for example the probability of this one non-terminal expansion in the tree is computed as follows:

$$\begin{aligned} P(\text{airline, flight indicator, flt-num, origin, destination}|\text{flight}) \; = \\ P(\text{flight indicator}|\text{airline, flight}) \\ \times P(\text{flt-num}|\text{flight indicator, flight}) \\ \times P(\text{origin}|\text{flt-num, flight})\ldots \end{aligned} \tag{19.6}$$

The two components of the HUM model are both trained from a labeled training set, just as the HMM model is. The semantic prior $P(C)$ is generated from a probabilistic finite-state network (a recursive transition network, or RTN) of concepts. Figure 19.7 shows one of the subnetworks for the ATIS *flight* concept; the *Flight* frame probabilistically generates a sequence of slots (date, origin, airline etc). The arcs on this network represent the (bigram) transition probability of one slot following another. The individual nodes like *airline* act as recursive *jump* arcs in the ATN, calling a subnetwork for the airline concept.

As with the HMM, decoding (choosing the most likely sequence of concepts for a given sentence) can be done via the Viterbi algorithm. Since the network

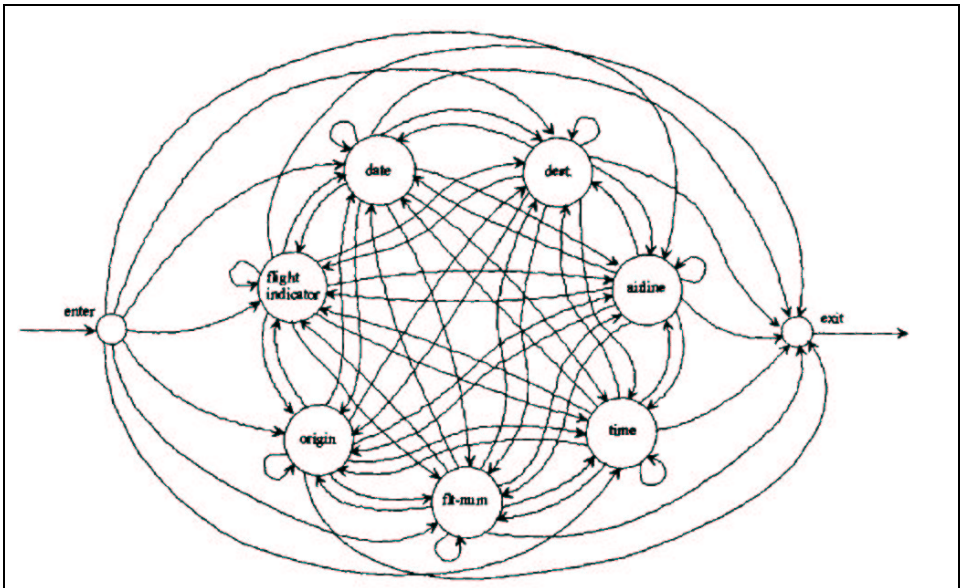

Figure 19.7 The computation of $P(C)$ from the Probabilistic RTN corresponding to the Flight concept, from (Miller et al., 1994). PLACEHOLDER FIGURE.

is a recursive transition network, states must be generated dynamically during the search.

As Young (2002) points out, one improvement of the hierarchical HUM model in Figure 19.6 over the flat HMM model in Figure 19.5 is that having preterminal labels (like *Origin* or *Dest*) avoids fragmenting the training data. In Figure 19.5, cities are split between being *Origins* and *Destinations*. In Figure 19.6, the class *city name* captures both kinds of cities, and facts about how the cities fit into larger structure is captured by the preterminal nodes.

Further complexity can be added to the HUM model by including syntactic as well as semantic knowledge into the grammar rules, essentially combining the insights of semantic grammars and probabilistic parsers. The TINA system (Seneff, 1995) is such a system; Figure 19.8 shows an example of a syntactic/semantic parse tree.

The node probabilities in TINA are generated by computing separate N-gram grammars for each non-terminal, conditioned on the parent non-terminal, and training on a hand-labeled training set.

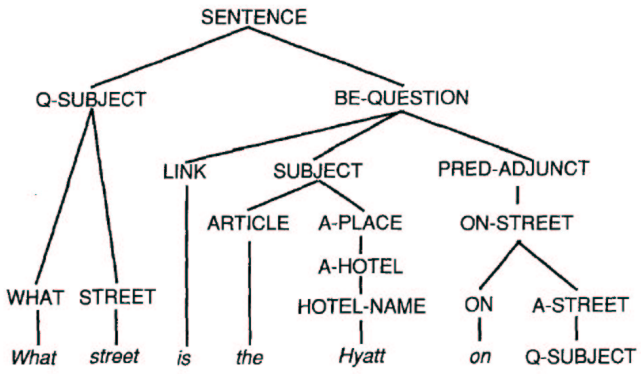

Figure 19.8 A parse of a sentence in the TINA semantic grammar (from (Seneff, 1995)). PLACEHOLDER FIGURE.

Generation and TTS components

The generation component of a conversational agent chooses the concepts to express to the user, plans out how to express these concepts in words, and assigns any necessary prosody to the words, as described in Chapter 20. The TTS component then takes these words and their prosodic annotations and synthesizes a waveform, as described in Chapter 30. Both these components may be optimized in various ways for use in conversational agents.

As Chapter 20 describes, the generation task can be separated into two tasks: *what to say*, and *how to say it*. The **content planner** module addresses the first task, decides what content to express to the user, whether to ask a question, present an answer, and so on. The content planning component of dialogue systems is often merged with the dialogue manager.

The **language generation** module addresses the second task, choosing the syntactic structures and words needed to express the meaning. Language generation modules are implemented in one of two ways. In the simplest and most common method, all or most of the words in the sentence to be uttered to the user are prespecified by the dialogue designer. This method is known as template-based generation. While most of the words in the template are fixed, templates can include some variables which are filled in by the generator, as in the following:

What time do you want to leave CITY-ORIG?
Will you return to CITY-ORIG from CITY-DEST?

A second method for language generation relies on the **natural language generation** techniques covered in Chapter 20. Here the dialogue manager builds a representation of the meaning of the utterance to be expressed, and passes this

meaning representation to a full generator. Such generators generally have three components, a sentence planner, surface realizer, and prosody assigner. A sketch of this architecture is shown in Figure 19.9.

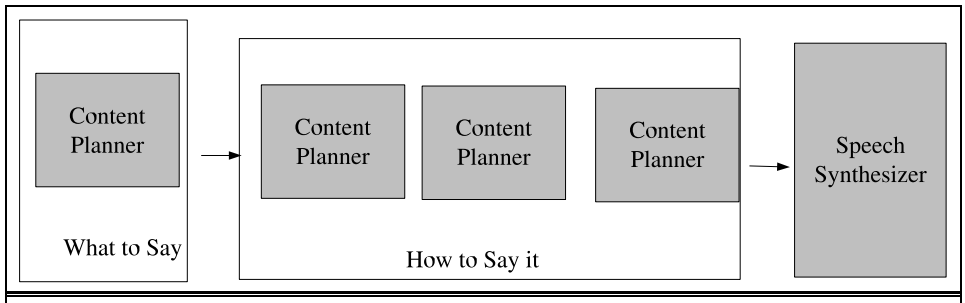

Figure 19.9 Architecture of a generator for a dialogue system, after Walker and Rambow (2002). PLACEHOLDER FIGURE. LABELS ARE WRONG IN MIDDLE BOX.

Whichever method is used, conversational dialogue places a number of constraints on the sentence generator related to Human Computer Interaction (HCI). Some of these constraints are not that different than other kinds of generation, and reflect the kind of discourse coherence discussed in Chapter 18. For example, as Cohen et al. (2004) show, the use of discourse markers and pronouns makes the dialogue in (19.8) more natural than the dialogue in (19.7):

(19.7) Please say the data.
...
Please say the start time.
...
Please say the duration.
...
Please say the subject.

(19.8) First, tell me the date.
...
Next, I'll need the time it starts.
...
Thanks. <pause> Now, how long is it supposed to last?
...
Last of all, I just need a brief description...

Another important case of discourse coherence occurs when particular prompts may need to be said to the user repeatedly. In these cases, it is standard in dialogue systems to use **tapered prompts**, prompts which get incrementally shorter. The following example from Cohen et al. (2004) shows a series of tapered prompts:

TAPERED PROMPTS

(19.9) System: Now, what's the first company to add to your watch list?

Caller: Cisco
System: What's the next company name? (Or, you can say, "Finished.")
Caller: IBM
System: Tell me the next company name, or say, "Finished."
Caller: Intel
System: Next one?
Caller: America Online.
System: Next?
Caller: ...

Other constraints on generation are more specific to spoken dialogue, and refer to facts about human memory and attentional processes. For example, when humans are prompted to give a particular response, it taxes their memory less if the suggested response is the last thing they hear. Thus as Cohen et al. (2004) point out, the prompt "To hear the list again, say 'Repeat list'" is easier for users than "Say 'Repeat list' to hear the list again."

Similarly, presentation of long lists of query results (e.g., potential flights, or movies) can tax users. Thus most dialogue systems have content planning rules to deal with this. In the Mercury system for travel planning described in (?), for example, a content planning rule specifies that if there are more than three flights to describe to the user, the system will just list the available airlines and describe explicitly only the earliest flight.

FIX: Add more here on Stent Prasad Walker '04 and Walker et al '03 CogSci

Dialogue Manager

The final component of a dialogue system is the dialogue manager, which controls the architecture and structure of the dialogue. The dialogue manager takes input from the ASR/NLU components, maintains some sort of state, interfaces with the task manager, and passes output to the NLG/TTS modules.

We saw a very simple dialogue manager in Chapter 2, where we introduced ELIZA. Recall that the architecture of ELIZA's was a simple read-substitute-print loop. The system read in a sentence, applied a series of text transformations to the sentence, and then printed it out. No state was kept; the transformation rules were only aware of the current input sentence. What makes a modern dialogue manager very different than ELIZA is both amount of state that the manager keeps about the conversation, and the ability of the manager to model structures of dialogue above the level of a single response.

Four kinds of dialogue management architectures are most common. Two are discussed here (finite-state and frame-based), and two are discussed in the Ad-

vanced Topics part of the chapter (plan-based architectures and Markov Decision Process models). We will also discuss a more advance dialogue architecture, the BDI architecture, in Section 19.9.

The simplest dialogue manager architecture is a finite-state manager. For example, imagine a trivial airline travel system whose job was to ask the user for a departure city, a destination city, a time, and whether the trip was round-trip or not. Figure 19.10 shows a sample dialogue manager for such a system. The states of the FSA correspond to questions that the dialogue manager asks the user, and the arcs correspond to actions to take depending on what the user responds.

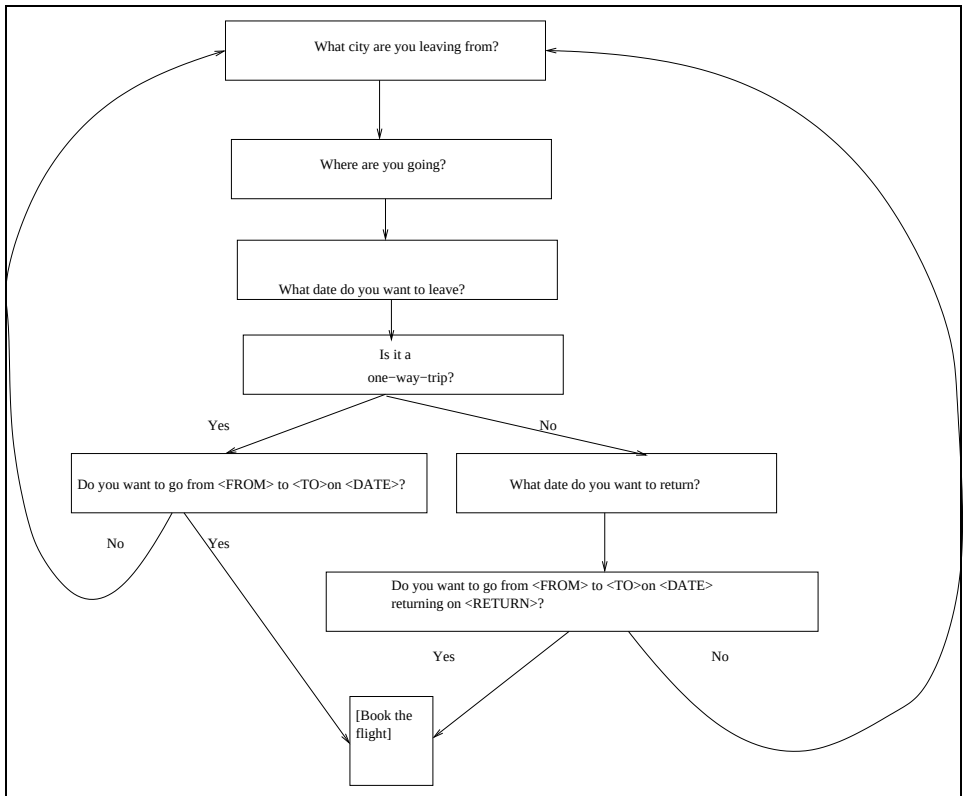

Figure 19.10 A simple finite-state automaton architecture for a dialogue manager. PLACEHOLDER FIGURE.

This system completely controls the conversation with the user. It asks the user a series of questions, ignoring (or misinterpreting) anything the user says that is not a direct answer to the system's question, and then going on to the next question.

Systems that control the conversation in this way are called **system initia-**

tive or **single initiative** systems. We say that the speaker that is in control of the conversation has the **initiative**. In normal human-human dialogue, initiative shifts back and forth between the participants (Walker and Whittaker, 1990). The limited single-initiative finite-state dialogue manager architectures may be sufficient for very simple tasks (perhaps for entering a credit card number, or a name and password, on the phone). Furthermore, they have the advantage that the system always knows what question the user is answering. This means the system can prepare the speech recognition engine with a specific language model tuned to answers for this question. Knowing what the user is going to be talking about also makes the task of the natural language understanding engine easier. Pure system-initiative finite-state dialogue manager architectures are probably too restricted, however, even for the relatively uncomplicated task of a spoken dialogue travel agent system.

SYSTEM INITIATIVE
SINGLE INITIATIVE
INITIATIVE

Single initiative systems can also be controlled by the user, in which case they are called **user initiative** systems. Pure user initiative systems are generally used for stateless database querying systems, where the user asks single questions of the system, which the system converts into SQL database queries, and returns the results from some database.

USER INITIATIVE

The problem is that neither of these kinds of single-initiative systems is practical for the majority of problems. Pure system-initiative systems require that the user answer exactly the question that the system asked. But this can make a dialogue awkward and annoying. Users often need to be able to say something that is not exactly the answer to a single question from the system. For example, in a travel planning situation, users often want to express their travel goals with complex sentences that may answer more than one question at a time, as in Communicator example (19.10) repeated from Figure 19.1, or ATIS example (19.11).

(19.10) Hi I'd like to fly to Seattle Tuesday morning

(19.11) I want a flight from Milwaukee to Orlando one way leaving after five p.m. on Wednesday.

A finite state dialogue system, as typically implemented, can't handle these kinds of utterances since it requires that the user answer each question as it is asked. Of course it is theoretically possible to create a finite state architecture which has a separate state for each possible subset of questions that the user's statement could be answering, but this would require a vast explosion in the number of states, making this a difficult architecture to conceptualize.

Most finite-systems do allow the user to do things other than answer exactly the question which the system asked. The systems allow **universal** commands. Universals are commands that can be said anywhere in the dialog. They are implemented by essentially allowing every state to recognized the universal commands in addition to the answer to the question that the system just asked. Common uni-

UNIVERSAL

versals include **help**, which gives the user a (possibly state-specific) help message, **start over** (or **main menu**), which returns the user to some specified main start state, and some sort of command to correct the system's understanding of the users last statement. For example, in the travel system of San-Segundo et al. (2001), when the system misrecognizes a user's utterance, the user can say **correct** and the system will erase the misrecognition and go back.

But adding universals to a system-initiative architecture is still insufficient. Therefore, most systems avoid the pure system-initiative finite-state approach and use an architecture that allows **mixed initiative**, in which conversational initiative can shift between the system and user at various points in the dialogue.

MIXED INITIATIVE

One common mixed initiative dialogue architecture relies on the structure of the frame itself to guide the dialogue. These **frame-based** or **form-based** dialogue managers asks the user questions to fill slots in the frame, but allow the user to guide the dialogue by giving information that fills other slots in the frame. Each slot in the frame may be associated with a question to ask the user, of the following type:

FRAME-BASED

FORM-BASED

Slot	**Question**
ORIGIN CITY	"From what city are you leaving?"
DESTINATION CITY	"Where are you going?"
DEPARTURE TIME	"When would you like to leave?"
ARRIVAL TIME	"When do you want to arrive?"

A frame-based dialogue manager thus needs to ask questions of the user, filling any slot that the user specifies, until it has enough information to perform a data base query, and then return the result to the user. If the user happens to answer two or three questions at a time, the system has to fill in these slots and then remember not to ask the user the associated questions for the slots. Not every slot need have an associated question, since the dialogue designer may not want the user deluged with questions. Nonetheless, the system must be able to fill these slots if the user happens to specify them. This kind of form-filling dialogue manager thus does away with the strict constraints that the finite-state manager imposes on the order that the user can specify information.

While some domains may be representable with a single frame, others, like the travel domain, seem to require the ability to deal with multiple frames. For example, once a frame-based system has performed a query looking for flights, there is likely to be more than one flight which meet the user's constraints. This means that the user will be given a list of choices, A frame-based system might need another kind of frame which has slots for identifying elements of lists of flights (*How much is the first one?* or *Is the second one non-stop?*). Other frames might have general route information (for questions like *Which airlines fly from*

Boston to San Francisco?), information about airfare practices (for questions like *Do I have to stay a specific number of days to get a decent airfare?*) or about car or hotel reservations. Since users may switch from frame to frame, and since they may answer a future question instead of the one the system asked, the system must be able to disambiguate which slot of which frame a given input is supposed to fill, and then switch dialogue control to that frame. A frame-based system is thus essentially a production rule system. Different types of inputs cause different productions to fire, each of which can flexibly fill in different frames. The production rules can then switch control based on factors such as the user's input and some simple dialogue history like the last question that the system asked. **FIX: Add a sentence here on Mercury production rules and Galaxy architecture**.

The frame-based or production-rule dialogue manager architecture thus is appropriate when the set of possible actions the user could want to take is relatively limited, but where the user might want to switch around a bit among these things.

Now that we've seen the frame-based architecture, let's return to our discussion of conversational initiative. It's possible in the same agent to allow system-initiative, user-initiative, and mixed-initiative interactions. We said earlier that initiative refers to who has control of the conversation at any point. The phrase **mixed initiative** is generally used in two ways. It can mean that the system or the user could arbitrarily take or give up the initiative in various ways (Walker and Whittaker, 1990; Chu-Carroll and Brown, 1997). This kind of mixed initiative is generally only possible in the advanced BDI kinds of dialogue systems described in Section 19.9. In form-based dialogue system, the term mixed initiative is used for a more limited kind of shift, operationalized based on a combination of prompt type (open versus directive) and the type of grammar used in the ASR. An **open prompt** is one in which the system gives the user very few constraints, allowing the user to respond however they please, as in:

OPEN PROMPT

How may I help you?

A **directive prompt** is one which explicitly instructs the user how to respond, as in:

DIRECTIVE PROMPT

Say *yes* if you accept the call; otherwise, say *no*.

In Section 19.1 we defined a **restrictive** grammar as a language model which strongly constrains the ASR system, only recognizing proper responses to a given prompt.

In Figure 19.11 we then give the definition of initiative used in form-based dialogue systems, following Singh et al. (2002) and others. Here a system initiative interaction uses a directive prompt and a restrictive grammar; the user is told how to respond, and the ASR system is constrained to only recognize the responses that are prompted for. In user initiative, the user is given an open prompt, and the grammar

	Prompt Type	
Grammar	**Open**	**Directive**
Restrictive	*Doesn't make sense*	System Initiative
Non-Restrictive	User Initiative	Mixed Initiative

Figure 19.11 A standard operational definition of initiative, following following Singh et al. (2002) and others.

must recognize any kind of response, since the user could say anything. Finally, in a mixed initiative interaction, the system gives the user a directive prompt with particular suggestions for response, but the non-restrictive grammar allows the user to respond outside the scope of the prompt.

Defining initiative as a property of the prompt and grammar type in this way allows systems to dynamically change their initiative type for different users and interactions. Novice users, or users with high speech recognition error, might be better served by more system initiative. Expert users, or those who happen to speak more recognizably, might do well with mixed or user initiative interactions. We will see later how machine learning techniques can be used to choose initiative.

We will return to more advanced dialogue manager architectures in Section 19.9.

19.2 VOICEXML

VOICEXML
VXML

VoiceXML is the Voice Extensible Markup Language, an XML-based dialogue design language released by the W3C. The goal of VoiceXML (henceforth **vxml**) is to create simple audio dialogues of the type we have been describing, making use of ASR and TTS, and dealing with very simple mixed-initiative in a frame-based architecture. While vxml is more common in the commercial rather than academic setting, it offers a convenient summary of the dialogue system design issues we have discussed, and will continue to discuss.

A vxml document contains a set of dialogs, each of which can be a *form* or a *menu*. We will limit ourselves to introducing forms; see (?) for more information on vxml in general. The VXML document in Figure 19.12 defines a form with a single field named 'transporttype'. The field has an attached prompt, *Please choose airline, hotel, or rental car*, which can be passed to the TTS system. It also has a grammar (language model) which is passed to the speech recognition engine to specify which words the recognizer is allowed to recognize. In the example in Figure 19.12, the grammar consists of a disjunction of the three words *airline*, *hotel*, and *rental car*.

```
<form>
  <field name="transporttype">
     <prompt>
        Please choose airline, hotel, or rental car.
     </prompt>
     <grammar type="application/x=nuance-gsl">
       [airline hotel "rental car"]
     </grammar>
  </field>
  <block>
    <prompt>
     You have chosen <value expr="transporttype">.
    </prompt>
  </block>
</form>
```

Figure 19.12 A minimal VXML script for a form with a single field. User is prompted, and the response is then repeated back.

A `<form>` generally consists of a sequence of `<field>`s, together with a few other commands. The example below has one field. Each field has a name (the name of the field below is `transporttype`) which is also the name of the variable where the user's response will be stored. The prompt associated with the field is specified via the `<prompt>` command. The grammar associated with the field is specified via the `<grammar>` command. VoiceXML supports various ways of specifying a grammar, including XML Speech Grammar, ABNF, and various commercial standards, like Nuance GSL. We will be using the Nuance GSL format in the following examples.

The VoiceXML interpreter walks through a VXML form in document order, repeatedly selecting each item in the form. If there are multiple fields, the interpreter will visit each one in order. The interpretation order can be changed in various ways, as we will see later. The example in Figure 19.13 shows a form with three fields, for specifying the origin, destination, and flight date of an airline flight.

The prologue of the example shows two global defaults for error handling. If the user doesn't answer after a prompt (i.e., silence exceeds a timeout threshold), the VoiceXML interpreter will play the `<noinput>` prompt. If the users says something, but it doesn't match the grammar for that field, the VoiceXML interpreter will play the `<nomatch>` prompt. After any failure of this type, it is normal to re-ask the user the question that failed to get a response. Since these routines can be called from any field, and hence the exact prompt will be different every time, VoiceXML provides a `<reprompt\>` command, which will repeat the prompt for whatever field caused the error.

The three fields of this form show another feature of VoiceXML, the `<filled>`

```
<noinput>
I'm sorry, I didn't hear you.  <reprompt/>
</noinput>

<nomatch>
I'm sorry, I didn't understand that.  <reprompt/>
</nomatch>

<form>
  <block>    Welcome to the air travel consultant.  </block>
  <field name="origin">
      <prompt>     Which city do you want to leave from?  </prompt>
      <grammar type="application/x=nuance-gsl">
        [(san francisco) denver (new york) barcelona]
      </grammar>
      <filled>
         <prompt>  OK, from <value expr="origin">  </prompt>
      </filled>
  </field>
  <field name="destination">
      <prompt>  And which city do you want to go to?   </prompt>
      <grammar type="application/x=nuance-gsl">
        [(san francisco) denver (new york) barcelona]
      </grammar>
      <filled>
         <prompt>   OK, to <value expr="destination">   </prompt>
      </filled>
  </field>
  <field name="departdate" type="date">
      <prompt>  And what date do you want to leave?  </prompt>
      <filled>
         <prompt>   OK, on <value expr="departdate">   </prompt>
      </filled>
  </field>
  <block>
      <prompt> OK, I have you are departing from  <value expr="origin">
              to <value expr="destination"> on <value expr="departdate">
      </prompt>
      send the info to book a flight...
  </block>
</form>
```

Figure 19.13 A VXML script for a form with 3 fields, which confirms each field and handles the `noinput` and `nomatch` situations.

tag. The `<filled>` tag for a field is executed by the interpreter as soon as the field has been filled by the user. Here, this feature is used to give the user a confirmation of their input.

The last field, `departdate`, shows another feature of VoiceXML, the `type` attribute. VoiceXML 2.0 specifies seven built-in grammar types, `boolean`, `currency`, `date`, `digits`, `number`, `phone`, and `time`. Since the type of this field is `date`, the speech recognizer will be automatically passed a language model (grammar) which only allows dates, and we don't need to specify the grammar here explicitly.

Figure 19.14 gives a final example which shows mixed initiative. In a mixed

```
<noinput>    I'm sorry, I didn't hear you.  <reprompt/>  </noinput>

<nomatch> I'm sorry, I didn't understand that.  <reprompt/> </nomatch>

<form>
   <grammar type="application/x=nuance-gsl">
    <![ CDATA[
    Flight (  ?[
              (i [wanna (want to)] [fly go])
              (i'd like to [fly go])
              ([(i wanna)(i'd like a)] flight)
           ]
           [
             ( [from leaving departing] City:x) {<origin $x>}
             ( [(?going to)(arriving in)] City:x) {<destination $x>}
             ( [from leaving departing] City:x
               [(?going to)(arriving in)] City:y) {<origin $x> <destination $y>}
           ]
           ?please
     )
     City [ [(san francisco) (s f o)] {return( "san francisco, california")}
            [(denver) (d e n)] {return( "denver, colorado")}
            [(seattle) (s t x)] {return( "seattle, washington")}
     ]
     ]]> </grammar>

  <initial name="init">
     <prompt> Welcome to the air travel consultant. What are your travel plans?  </prompt>
  </initial>

  <field name="origin">
     <prompt> Which city do you want to leave from?  </prompt>
     <filled>
        <prompt> OK, from <value expr="origin"> </prompt>
     </filled>
  </field>
  <field name="destination">
     <prompt> And which city do you want to go to?  </prompt>
     <filled>
        <prompt> OK, to <value expr="destination"> </prompt>
     </filled>
  </field>
  <block>
     <prompt> OK, I have you are departing from  <value expr="origin">
              to  <value expr="destination">.  </prompt>
    send the info to book a flight...
  </block>
</form>
```

Figure 19.14 A mixed initiative VXML dialog. The grammar allows sentences which specify the origin or destination cities or both. User can respond to the initial prompt by specifying origin city, destination city, or both.

initiative dialogue, users can choose not to answer the question that was asked by the system. For example, they might answer a different question, or use a long sentence to fill in multiple slots at once. This means that the VoiceXML interpreter

can no longer just evaluate each field of the form in order; it needs to skip fields whose values are set. This is done by a *guard condition*, a test that keeps a field from being visited. The default guard condition for a field tests to see if the field's form item variable has a value, and if so the field is not interpreted.

Figure 19.14 also shows a much more complex use of a grammar. This grammar is a CFG grammar with two rewrite rules, named `Flight` and `City`. The Nuance GSL grammar formalism uses parentheses () to mean concatenation and square brackets [] to mean disjunction. Thus a rule like (19.12) means that `Wantsentence` can be expanded as `i want to fly` or `i want to go`, and `Airports` can be expanded as `san francisco` or `denver`.

(19.12)
```
Wantsentence (i want to [fly go])
    Airports [(san francisco) denver]
```

Grammar rules can refer to other grammar rules recursively, and so in the grammar in Figure 19.14 we see the grammar for `Flight` referring to the rule for `City`. VoiceXML grammars take the form of CFG grammars with optional semantic attachments. The semantic attachments for the `City` rule passes up the city and state name as its semantics. The semantic attachments for the `Flight` rule takes this value and pass it up filling in the correct slot (`<origin>` or `<destination>`). Because Figure 19.14 is a mixed initiative grammar, the grammar has to be applicable to any of the fields. This is done by making the expansion for `Flight` be a disjunction; note that it allows the user to specify only the origin city, only the destination city, or both.

FIX: ADD A PGRAPH ON SEMANTIC ATTACHMENTS TO GRAMMAR RULES IN VXML AND CONCLUDE

19.3 DIALOGUE SYSTEM EVALUATION

An optimal dialogue system is one which allows a user to accomplish their goals (maximizing task success) with the least problems (minimizing costs). A number of metrics for each of these two criteria have been proposed.

Task Completion Success: Task success can be measured by evaluating the correctness of the total solution. For a frame-based architecture, this might be the percentage of slots that were filled with the correct values, or the percentage of subtasks that were completed (Polifroni et al., 1992). Since different dialogue systems may be applied to different tasks, it is hard to compare them on this metric, so Walker et al. (1997) suggested using the Kappa coefficient, κ, to compute a completion score which is normalized for chance agreement and better enables cross-system comparison.

METHODOLOGY BOX: DESIGNING DIALOGUE SYSTEMS

How does a dialogue system developer choose dialogue strategies, architectures, prompts, error messages, and so on? The three design principles of Gould and Lewis (1985) can be summarized as: **User-Centered Design:** Study the user and task, Build simulations and prototypes, and Iteratively test them on the user and fix the problems.

1. Early Focus on Users and Task: Understand the potential users and the nature of the task, via interviews with users and investigation of similar systems. Study of related human-human dialogues can also be useful, although the language in human-machine dialogues is usually simpler than in human-human dialogues.

2. Build Prototypes: In Wizard-of-Oz (WOZ) or PNAMBIC (Pay No Attention to the Man BehInd the Curtain) systems, the users interact with what they think is a software system, but is in fact a human operator ("wizard") behind some disguising interface software (e.g. Gould et al., 1983; Good et al., 1984; Fraser and Gilbert, 1991) . (The name comes from the children's book *The Wizard of Oz* (Baum, 1900), in which the Wizard turned out to be just a simulation controlled by a man behind a curtain.) A WOZ system can be used to test out an architecture without implementing the complete system; only the interface software and databases need to be in place. The wizard's linguistic output can be be disguised by a text-to-speech system. It is difficult for the wizard to exactly simulate the errors, limitations, or time constraints of a real system; results of WOZ studies are thus somewhat idealized.

3. Iterative Design: An iterative design cycle with embedded user testing is essential in system design (Nielsen, 1992; Cole et al., 1994, 1997; Yankelovich et al., 1995; Landauer, 1995). For example Stifelman et al. (1993) and Yankelovich et al. (1995) found that users of speech systems consistently tried to interrupt the system (**barge in**), suggesting a redesign of the system to recognize overlapped speech. Kamm (1994) and Cole et al. (1993) found that **directive prompts** ("Say *yes* if you accept the call, otherwise, say *no*") or the use of constrained forms (Oviatt et al., 1993) produced better results than open prompts like "Will you accept the call?".

Summary of Cohen's recommendations on prompt design, tapering, etc Summary of Cohen's recommendations on errors

Finally, Walker et al. (2001) notes that sometimes the user's *perception* of whether they completed the task is a better predictor of user satisfaction than the above measures. In more recent studies on evaluation of dialogue systems, Walker et al. (2002) gives users an on-line survey after completing a dialogue, which ask for a yes-no answer as to whether the task was completed.

Efficiency Cost: Efficiency costs are measures of the system's efficiency at helping users. This can be measured via the total elapsed time for the dialogue in seconds, the number of total turns or of system turns, or the total number of queries (Polifroni et al., 1992). Other metrics include the number of system non-responses, and the "turn correction ratio": the number of system or user turns that were used solely to correct errors, divided by the total number of turns (Danieli and Gerbino, 1995; Hirschman and Pao, 1993).

Quality Cost: Quality cost measures other aspects of the interaction that affect users' perception of the system. One such measure is the number of times the ASR system failed to return any sentence, or the number of ASR rejection prompts ('I'm sorry, I didn't understand that'). Similar metrics include the number of times the user had to barge-in to the system, or the number of time-out prompts played when the user didn't respond quickly enough. A number of quality metrics deal with how well the system understood and responded to the user. This can include the inappropriateness (whether verbose or ambiguous) of the system's questions, answers, and error messages (Zue et al., 1989), or the correctness (or partial correctness) of each question, answer, or error message (Zue et al., 1989; Polifroni et al., 1992).

CONCEPT ACCURACY

A very important quality cost is **concept accuracy** or **concept error rate**, which measures the percentage of semantic concepts that the NLU component returns correctly. For frame-based architectures this can be measured by counting the percentage of slots that are filled with the correct meaning. For example if the sentence 'I want to arrive in Austin at 5:00' is misrecognized to have the semantics "DEST-CITY: Boston, Time: 5:00" the concept accuracy would be 50% (one of two slots are wrong) (?).

How should these success and cost metrics be combined and weighted? The PARADISE algorithm (Walker et al., 1997) (PARAdigm for DIalogue System Evaluation) applies multiple regression to this problem. The algorithm first uses questionnaires to assign each dialogue a user satisfaction rating. A set of cost and success factors like those above is then treated as a set of independent factors; multiple regression is used to train a weight (coefficient) for each factor, measuring its importance in accounting for user satisfaction. Figure 19.15 shows the particular model of performance that the PARADISE experiments have assumed. Each box is related to a set of factors that we summarized on the previous page. The resulting metric can be used to compare quite different dialogue strategies.

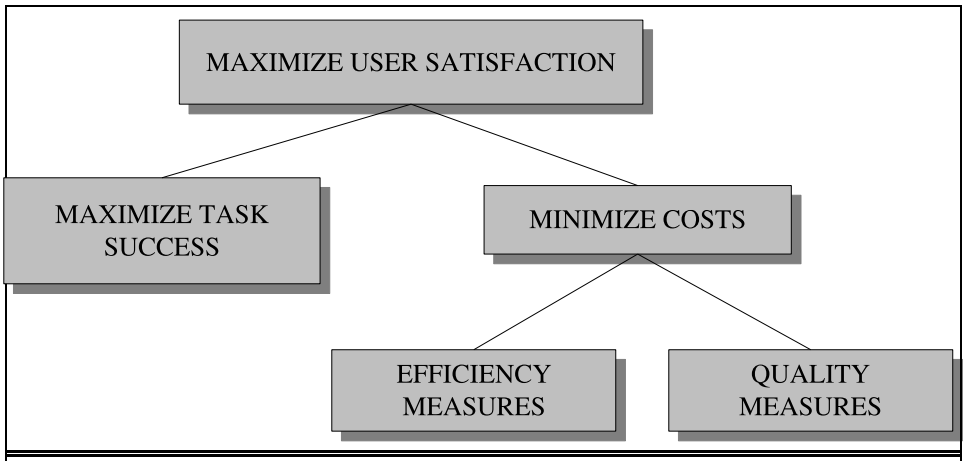

Figure 19.15 PARADISE's structure of objectives for spoken dialogue performance. After (Walker et al., 2001).

The user satisfaction rating is computed by having users complete a survey with questions such as those in Figure 19.16, probing their perception of different aspects of the system's performance (Shriberg et al., 1992; Polifroni et al., 1992; Stifelman et al., 1993; Yankelovich et al., 1995). Surveys in PARADISE studies are multiple choice, with the responses mapped into the range of 1 to 5. The scores for each question are then averaged to get a total user satisfaction rating.

- Was the system easy to understand ? (**TTS Performance**)
- Did the system understand what you said? (**ASR Performance**)
- Was it easy to find the message/flight/train you wanted? (**Task Ease**)
- Was the pace of interaction with the system appropriate? (**Interaction Pace**)
- Did you know what you could say at each point of the dialogue? (**User Expertise**)
- How often was the system sluggish and slow to reply to you? (**System Resonse**)
- Did the system work the way you expected hjim to? (**Expected Behavior**)
- Do you think you'd use the system in the future? (**Future Use**)

Figure 19.16 User satisfaction survey, adapted from (Walker et al., 2001).

Walker et al. (2001, 2002) applied the PARADISE algorithm to three different dialogue systems and found that three factors that were often the best predictors of user satisfaction were (1) the average concept accuracy for the whole dialogue, (2) the user's (binary) opinion about whether they completed the task successfully, and (3) the total elapsed time.

19.4 GROUNDING, CONFIRMATION, AND ERRORS

The simple finite-state and forms-based systems we discussed in the previous section are only capable of handling relatively limited dialogues, dealing with only the simplest of the phenomena that characterize true fluent dialogues. In the next few sections, we will introduce a number of more sophisticated theoretical ideas and practical techniques, leading up to progressively more and more powerful dialogue systems. We begin in this section with the idea of grounding.

Grounding

Dialogue is a collective act performed by the speaker and the hearer. One implication of this collectiveness is that, unlike in monologue, the speaker and hearer must constantly establish **common ground** (Stalnaker, 1978), the set of things that are mutually believed by both speakers. The need to achieve common ground means that the hearer must **ground** or **acknowledge** the speaker's utterances, making it clear that the hearer has understood the speaker's meaning and intention. As Clark (1996) points out, people need closure on their actions; he phrases this as follows (after (Norman, 1988)):

COMMON GROUND

GROUND

ACKNOWLEDGE

> *Principle of closure.* Agents performing an action require evidence, sufficient for current purposes, that they have succeeded in performing it.

Grounding is also important when the hearer needs to indicate that the speaker has *not* succeeded in performing an action. If the hearer has problems in understanding, she must indicate these problems to the speaker, again so that mutual understanding can eventually be achieved.

Let's begin by discussing ways that the hearer (call her B) can use to ground the speaker A's utterance. Clark and Schaefer (1989) discuss five main types of methods, ordered from weakest to strongest:

1. **Continued attention:** B shows she is continuing to attend and therefore remains satisfied with A's presentation.
2. **Relevant next contribution:** B starts in on the next relevant contribution.
3. **Acknowledgement:** B nods or says a continuer like *uh-huh*, *yeah*, or the like, or an **assessment** like *that's great*.
4. **Demonstration:** B demonstrates all or part of what she has understood A to mean, for example by **reformulating** (paraphrasing) A's utterance, or by **collaborative completion** of A's utterance.
5. **Display:** B displays verbatim all or part of A's presentation.

REFORMULATING

COLLABORATIVE COMPLETION

Let's look for examples of these in a human-human dialogue example. We'll be returning to this example throughout the chapter; in order to design a more sophisticated machine dialogue agent, it helps to look at how a human agent performs similar tasks. Figure 19.17 shows part of a dialogue between a human travel agent and a human client.

C_1:	. . . I need to travel in May.
A_1:	And, what day in May did you want to travel?
C_2:	OK uh I need to be there for a meeting that's from the 12th to the 15th.
A_2:	And you're flying into what city?
C_3:	Seattle.
A_3:	And what time would you like to leave Pittsburgh?
C_4:	Uh hmm I don't think there's many options for non-stop.
A_4:	Right. There's three non-stops today.
C_5:	What are they?
A_5:	The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.
C_6:	OK I'll take the 5ish flight on the night before on the 11th.
A_6:	On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.
C_7:	OK.

Figure 19.17 A fragment from a telephone conversation between a client (C) and a travel agent (A).

Utterance A_1, in which the agent repeats *in May*, repeated below in boldface, shows the strongest form of grounding, in which the hearer displays their understanding by repeating verbatim part of the speakers words:

C_1: . . . I need to travel **in May**.
A_1: And, what day **in May** did you want to travel?

This particular fragment doesn't have an example of an *acknowledgement*, but there's an example in another fragment:

C: He wants to fly from Boston
A: **Mm hmm**
C: to Baltimore Washington International

The word *mm-hmm* here is a **continuer**, also often called a **backchannel** or an **acknowledgement token**. A continuer is a short utterance which acknowledges the previous utterance, showing the speaker that she is understand and hence cueing

CONTINUER
BACKCHANNEL

the other speaker to continue talking (Jefferson, 1984; Schegloff, 1982; Yngve, 1970).

In Clark and Schaefer's third method, the speaker starts in on their relevant next contribution. We see a number of examples of this in the sample dialogue above, for example where the speaker asks a question and the hearer answers it. In a more subtle but very important kind of grounding act, the speaker can combine this method with the previous one. For example notice that whenever the client answers a question, the agent begins the next question with *And*. The *And* indicates to the client that the agent has successfully understood the answer to the last question:

And, what day in May did you want to travel?
..
And you're flying into what city?
..
And what time would you like to leave Pittsburgh?

Grounding is not just an interesting tidbit about human conversation; it is important to the design of a conversation agent, and has played a large role in recent research. Conversational agents need to ground in order to behave in the way that humans expect of a conversational partner. Stifelman et al. (1993) and Yankelovich et al. (1995) found that users of speech-based interfaces are often confused when the system doesn't give them an explicit acknowledgement signal after processing the user's utterances. Cohen et al. (2004) gives a specific example of this, noting that it is important for dialogue systems to ground a user's negative response to a system's offer. For example, the use of *Okay* makes (19.13) a more natural response than (19.14):

(19.13) System: Did you want to review some more of your personal profile?
Caller: No.
System: *Okay*, what's next?

(19.14) System: Did you want to review some more of your personal profile?
Caller: No.
System: What's next?

To understand a second key function of grounding in conversational agents, we have to remember why humans ground in the first place: because for all its richness, language is a pretty thin channel for two independent minds full of information to send each other vast swaths of that information for complete understanding. Language is thus fraught with misunderstandings, mishearings, and miscommunication in general. Grounding is part of the process by which speakers deal with these problems so as to achieve mutual understanding.

Confirmation

In a dialogue system, mishearings are a particularly important class of problems, because speech recognition has such a high error rate. It is therefore important for dialogue systems to make sure that they have achieved the correct interpretation of the user's input. This is generally done by confirmation with the user. A system using the **explicit confirmation** strategy asks the user a direct question to confirm their understanding. Here are two examples of explicit confirmations from travel planning dialogue systems. The (boldface) confirmation questions are both yes-no questions, one using a single sentence, the other presenting a declarative sentence followed by a tag question (a short question phrase like "right?" or "isn't it?"):

EXPLICIT CONFIRMATION

S:	Which city do you want to leave from?
U:	Baltimore.
S:	**Do you want to leave from Baltimore?**
U:	Yes.

U:	I'd like to fly from Denver Colorado to New York City on September twenty first in the morning on United Airlines
S:	**Let's see then. I have you going from Denver Colorado to New York on September twenty first. Is that correct?**
U:	Yes

A system using the **implicit confirmation** strategy, rather than asking a direct question, uses the *demonstration* or *display* grounding strategies described above, confirming to the user by repeating back what the system understood the user to have said:

IMPLICIT CONFIRMATION

U:	I want to travel to Berlin
S:	**When do you want to travel to Berlin?**

U2:	Hi I'd like to fly to Seattle Tuesday Morning
A3:	**Traveling to Seattle on Tuesday, August eleventh in the morning.** Your full name?

Explicit and implicit confirmation have complementary strengths. Explicit confirmation makes it much easier for users to correct the system's misrecognitions since the user can simply answer 'no' to the confirmation question (Danieli and Gerbino, 1995; Krahmer et al., 1999). But explicit confirmation is awkward and increases the length of the conversation (Danieli and Gerbino, 1995; Walker et al., 1998). The explicit confirmation dialogue fragments above sound non-natural and definitely non-human; implicit confirmation is thus much more conversationally natural.

While early dialogue systems tended to chose either explicit or implicit confirmation, recent systems are more adaptive, changing their confirmation strategy

from sentence to sentence. Various factors can be used in making this decision. The most important factor is some measure of ASR performance. A number of systems, for example, use the acoustic confidence that the ASR system assigns to an utterance, computed from the acoustic log-likelihood of the utterance, to decide whether to make an explicit confirmation. Such systems explicitly confirm sentences for which the recognizer was not confident of its output (Bouwman et al., 1999; San-Segundo et al., 2001; Litman et al., 1999; Litman and Pan, 2002).Recent research has focused on more sophisticated measures of confidence that go beyond acoustic log-likelihood, such as prosodic factors; for example utterances with longer prior pauses, F0 excursions, and longer durations are likely to be misrecognized, (Litman et al., 2000). Another important factor in deciding whether to explicitly confirm is the cost of an error; obviously before actually booking a flight or moving money in an account, explicit confirmation is important (Kamm, 1994; Cohen et al., 2004). All of these factors can thus be combined in a machine learning approach to predict whether explicit confirmation should be used. Section 19.8 introduces the markov decision process (MDP) model of dialogue, in which reinforcement learning can be used to optimally choose strategies like explicit versus implicit confirmation.

Rejection

REJECTION

Confirmation is just one option that a system has to express lack of understanding. Another option is **rejection**. An ASR system rejects an utterance by giving the user a prompt like *I'm sorry, I didn't understand that*, as in the VXML `nomatch` prompts we saw in Section 19.2. Rejection might happen when the ASR confidence is so low, or the best interpretation is so semantically ill-formed, that the system can be relatively sure that the user's input was not recognized at all. Systems thus might have a three-tiered level of confidence; below a certain confidence threshold, an utterance is rejected. Above the thresheld, it is explicitly confirmed. If the confidence is even higher, the utterance is implicitly confirmed.

PROGRESSIVE PROMPTING

Sometimes utterances are rejected multiple times. This might mean that the user is using language that the system is unable to follow. Thus when an utterance is rejected, systems often follow a strategy of **progressive prompting** or **escalating detail** (Yankelovich et al., 1995; ?) as shown in the following example from Cohen et al. (2004):

System: When would you like to leave?
Caller: Well, um, I need to be in New York in time for the first World Series game.
System: <reject>. Sorry, I didn't get that. Please say the month and day you'd like to leave.
Caller: I wanna go on October fifteenth.

In this example, instead of just repeating 'When would you like to leave?', the rejection prompt gives the caller more guidance about how to formulate an utterance the system will understand. If the caller's utterance gets rejected yet again, the prompt can reflect this ('I *still* didn't get that'), and give the caller even more guidance. An alternative strategy for error handling is **rapid reprompting**, in which the system rejects an utterance just by saying "I'm sorry?" or "What was that?". Only if the caller's utterance is rejected a second time does the system start applying progressive prompting. Cohen et al. (2004) summarizes experiments showing that users greatly prefer rapid reprompting as a first-level error prompt.

RAPID REPROMPTING

Instead of rejecting or confirming entire utterances, it would be nice to be able to clarify only the parts of the utterance that the system didn't understand. If a system can assign confidence at a more fine-grained level than the utterance, it can clarify such individual elements via **clarification subdialogues**.

CLARIFICATION SUBDIALOGUES

Error handling

Despite all the cleverness we use in designing confirmation and rejection strategies, dialogue systems will still make mistakes. One kind of error occurs as a result of rejection or confirmation. If the system misrecognizes an utterance and either rejects or uses explicit or implicit confirmation mechanisms, the user has a chance to correct the error, perhaps by repeating themselves, or rephrasing the utterance, or by saying 'no' to the confirmation question, and so on. Such utterances are called **corrections**. The most common cause of corrections is ASR errors. Handling corrections is an important part of error handling. Dialogue systems need to recognize that users are doing a correction, and then figure out what the user is trying to correct, perhaps by interacting with the user further.

CORRECTIONS

Unfortunately, corrections are actually *harder* to recognize than normal sentences. (Swerts et al., 2000) found that corrections in the TOOT dialogue system were misrecognized about twice as often (in terms of WER) as non-corrections. One reason for this is that speakers use a very different prosodic style for corrections, a style called **hyperarticulation**. In hyperarticulated speech, some part of the utterance has exaggerated energy, duration, or F0 contours, such as *I said BAL-TI-MORE, not Boston* (?; Oviatt et al., 1998; Levow, 1998; Hirschberg et al., 2001). Besides these prosodic facts, what are the lexical and syntactic characteris-

HYPERARTICULATION

tics of corrections? Krahmer et al. (1999) and Levow (1998) found that corrections were generally longer than non-corrections. (Swerts et al., 2000) characterize five kinds of strategies users use: repetition (REP), paraphrase (PAR), add task relevant content (ADD), omit content (OMIT), and both add and omit content. An example dialogue is shown in Figure 19.18. (Swerts et al., 2000) found that about two-third of all corrections were repetitions and omissions; paraphrase was less likely to be used.

	Turn	Corr Type
S	On which day of the week do you want to leave?	
S	Sorry, I didn't hear anything from you. Please tell me your departure date	
U	*Um tomorrow*	NA
S	You can say the departure date. For example, you can say 'today' or 'tomorrow'. Or you can say 'Monday', 'Tuesday', 'Wednesday'...	
U	*Tomorrow*	REP
S	You can say the departure date. For example, you can say 'today' or 'tomorrow'. Or you can say 'Monday', 'Tuesday', 'Wednesday'...	
U	*Saturday*	PAR
S	Do you want to leave on Saturday?	
U	Yes	
S	What time do you want to leave?	
U	Eight o'clock A M	
S	Do you want to leave around ten oclock p-m	
U	*Eight o'clock*	OMIT
S	Do you want to leave around ten oclock p-m	
U	*No*	OMIT
S	What time do you want to leave?	
U	*Eight o'clock A M*	ADD

Figure 19.18 Dialogue fragment from the TOOT system labeled for correction type, from (Swerts et al., 2000).

We saw in the previous section that machine learning approaches could be used to automatically identify system errors such as misrecognitions. The same sorts of approaches can be used to automatically identify *user corrections* of these errors. Various researchers (Levow, 1998; Hirschberg et al., 2001; Bulyko et al., 2004) have built classifiers for deciding if an utterances was a correction. These classifiers use features like lexical information (words like "no", "correction", "I don't", swear words), prosodic features (various kinds of increases in F0 range, pause duration, and word duration features that correlate with hyperarticulation, generally normalized by the values for previous sentences), features indicating the

total length of the utterance in words or seconds, various ASR features (confidence, language model probability), and various dialogue features. (Bulyko et al., 2004) showed how this kind of error corrector could be integrated into a dialogue manager, adding the 'error correction subsystem' shown in Figure 19.19. The node labeled 'check for error correction' in this figure implements a correction-detector like those discussed above.

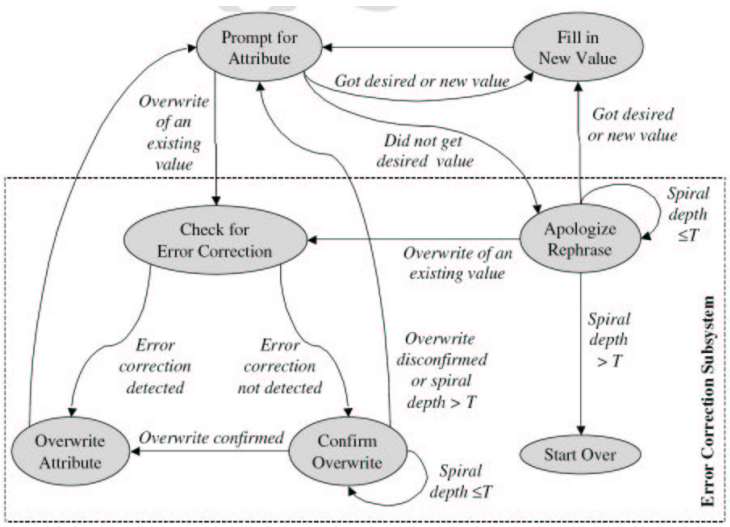

Figure 19.19 Error correction subsystem component of a dialogue manager, from (Bulyko et al., 2004). PLACEHOLDER FIGURE.

Corrections are not the only kind of error we need to handle in dialogue systems. In telephony applications such as call routing, we might use a human operator as a backup if the system fails. Predicting that a dialog is going to be problematic, so that we can hand the call over to a human operator, is another error handling task. Walker et al. (2000) studied three kinds of problematic call routing dialogues; ones where the user hangs up, ones where a monitoring human decided to step in and take over the call, and ones where the system routed the call incorrectly. They trained a RIPPER classifier on a wide variety of prosodic, NLU, and dialogue features, extracted from the first exchange (first two turns) from a dialogue. The best predictor of a problematic dialogue was if the NLU confidence score for the top-ranked interpretation was low and there was touchtone (DTMF) input in the user utterance. Another predictor for problematic dialogues was long utterances where the difference between the confidence scores of the top and second-ranked interpretations was low.

19.5 DIALOGUE ACTS

An important insight about conversation, due to Austin (1962), is that an utterance in a dialogue is a kind of **action** being performed by the speaker. This is particularly clear in **performative** sentences like the following:

PERFORMATIVE

(19.15) I name this ship the *Titanic*.

(19.16) I second that motion.

(19.17) I bet you five dollars it will snow tomorrow.

When uttered by the proper authority, for example, (19.15) has the effect of changing the state of the world (causing the ship to have the name *Titanic*) just as any action can change the state of the world. Verbs like *name* or *second* which perform this kind of action are called performative verbs, and Austin called these kinds of actions **speech acts**. What makes Austin's work so far-reaching is that speech acts are not confined to this small class of performative verbs. Austin's claim is that the utterance of any sentence in a real speech situation constitutes three kinds of acts:

SPEECH ACTS

- **locutionary act:** the utterance of a sentence with a particular meaning.
- **illocutionary act:** the act of asking, answering, promising, etc., in uttering a sentence.
- **perlocutionary act:** the (often intentional) production of certain effects upon the feelings, thoughts, or actions of the addressee in uttering a sentence.

For example, Austin explains that the utterance of example (19.18) might have the **illocutionary force** of protesting and the perlocutionary effect of stopping the addressee from doing something, or annoying the addressee.

ILLOCUTIONARY FORCE

(19.18) You can't do that.

The term **speech act** is generally used to describe illocutionary acts rather than either of the other two levels. Searle (1975b), in modifying a taxonomy of Austin's, suggests that all speech acts can be classified into one of five major classes:

- **Assertives:** committing the speaker to something's being the case (*suggesting*, *putting forward*, *swearing*, *boasting*, *concluding*).
- **Directives:** attempts by the speaker to get the addressee to do something (*asking*, *ordering*, *requesting*, *inviting*, *advising*, *begging*).
- **Commissives:** committing the speaker to some future course of action (*promising*, *planning*, *vowing*, *betting*, *opposing*).
- **Expressives:** expressing the psychological state of the speaker about a state of affairs *thanking*, *apologizing*, *welcoming*, *deploring*.

- **Declarations:** bringing about a different state of the world via the utterance (including many of the performative examples above; *I resign*, *You're fired*.)

While speech acts provide a useful characterization of one kind of pragmatic force, more recent work, especially computational work in building dialogue systems, has significantly expanded this core notion, modeling more kinds of conversational functions that an utterance can perform. The resulting enriched acts are often called **dialogue acts** (Bunt, 1994) or **conversational moves** (Power, 1979; Carletta et al., 1997b). The phrase 'dialogue act' is unfortunately ambiguous. As Bunt and Black (2000) point out, it has been variously used to loosely mean 'speech act, in the context of a dialogue' (Bunt, 1994), to mean a combination of the speech act and semantic force of an utterance (Bunt, 2000), or to mean an act with internal structure related specifically to its dialogue function (Allen and Core, 1997). The third usage is perhaps the most common in the literature, and we will mainly rely on it here.

DIALOGUE ACT
MOVES

Let's begin by looking at one set of dialogue acts that is used for a particular domain of task-oriented dialogue. The Verbmobil corpus consists of two-party scheduling dialogues, in which the speakers were asked to plan a meeting at some future date. This data was used to design conversational agents which would help with this task. A dialogue act tagset, shown in Figure 19.20, was designed to tag the dialogue function of each of the sentences in the corpus. Notice that it has many very domain-specific tags, such as SUGGEST, used for when someone proposes a particular date to meet, and ACCEPT and REJECT, used to accept or reject a proposal for a date.

The goal of this dialogue act tagset is to help build a conversational agent which could discuss meeting scheduling in a complex mixed-initiative way, allowing the user and the system to make proposals, accept and reject them, make counter-proposals, and in general behave in ways that would be hard to fit into the simple form-filling dialogues discussed in Section 19.1. In order to do this, the system has to be able to know when the user has asked a question, or whether the user has just rejected a proposal for a meeting on a particular date. Similarly, the system has to be able to use the proper dialogue act to make a proposal to the user, or ask a question. Thus dialogue act recognition and dialogue act planning are crucial steps in building more sophisticated agents.

Many dialogue act tagsets used for these purposes are quite domain-dependent. There is one effort to develop a more domain-independent dialogue act tagging scheme, the DAMSL (Dialogue Act Markup in Several Layers) architecture (Allen and Core, 1997; Walker et al., 1996; Carletta et al., 1997a; Core et al., 1999).

DAMSL draws inspiration from the work on grounding discussed in Section 19.4 (Clark and Schaefer, 1989; Clark, 1996), on **repair** (Schegloff et al.,

Tag	Example
THANK	*Thanks*
GREET	*Hello Dan*
INTRODUCE	*It's me again*
BYE	*Allright bye*
REQUEST-COMMENT	*How does that look?*
SUGGEST	*from thirteenth through seventeenth June*
REJECT	*No Friday I'm booked all day*
ACCEPT	*Saturday sounds fine,*
REQUEST-SUGGEST	*What is a good day of the week for you?*
INIT	*I wanted to make an appointment with you*
GIVE_REASON	*Because I have meetings all afternoon*
FEEDBACK	*Okay*
DELIBERATE	*Let me check my calendar here*
CONFIRM	*Okay, that would be wonderful*
CLARIFY	*Okay, do you mean Tuesday the 23rd?*
DIGRESS	*[we could meet for lunch] and eat lots of ice cream*
MOTIVATE	*We should go to visit our subsidiary in Munich*
GARBAGE	*Oops, I-*

Figure 19.20 The 18 high-level dialogue acts used in Verbmobil-1, abstracted over a total of 43 more specific dialogue acts. Examples are from Jekat et al. (1995).

1977), and on the relation of utterances to the preceding and succeeding discourse (Allwood et al., 1992; Allwood, 1995; Schegloff, 1968, 1988). For example, drawing on Clark and Allwood's work, the DAMSL tag set distinguishes between the **forward looking** and **backward looking** function of an utterance. The forward looking function of an utterance corresponds in many ways to the Searle/Austin speech act, although with a richer hierarchical structure (not discussed here) and more focus on task-oriented dialogue:

Forward Looking Function	
STATEMENT	a claim made by the speaker
INFO-REQUEST	a question by the speaker
CHECK	a question for confirming information
INFLUENCE-ON-ADDRESSEE	(=Searle's directives)
OPEN-OPTION	a weak suggestion or listing of options
ACTION-DIRECTIVE	an actual command
INFLUENCE-ON-SPEAKER	(=Austin's commissives)
OFFER	speaker offers to do something, (subject to confirmation)
COMMIT	speaker is committed to doing something
CONVENTIONAL	other
OPENING	greetings
CLOSING	farewells
THANKING	thanking and responding to thanks

The backward looking function of DAMSL focuses on the relationship of an utterance to previous utterances by the other speaker. These include accepting and rejecting proposals (since DAMSL is focused on task-oriented dialogue), and grounding and repair acts:

Backward Looking Function	
AGREEMENT	speaker's response to previous proposal
ACCEPT	accepting the proposal
ACCEPT-PART	accepting some part of the proposal
MAYBE	neither accepting nor rejecting the proposal
REJECT-PART	rejecting some part of the proposal
REJECT	rejecting the proposal
HOLD	putting off response, usually via subdialogue
ANSWER	answering a question
UNDERSTANDING	whether speaker understood previous
SIGNAL-NON-UNDER.	speaker didn't understand
SIGNAL-UNDER.	speaker did understand
ACK	demonstrated via continuer or assessment
REPEAT-REPHRASE	demonstrated via repetition or reformulation
COMPLETION	demonstrated via collaborative completion

Figure 19.21 shows a labeling of our sample conversation using versions of the DAMSL Forward and Backward tags.

19.6 Automatic Interpretation of Dialogue Acts

The previous section introduced dialogue acts and other activities that utterances can perform. This section turns to the problem of identifying or interpreting these acts. That is, how do we decide whether a given input is a QUESTION, a STATEMENT, a SUGGEST (directive), or an ACKNOWLEDGEMENT?

At first glance, this problem looks simple. We saw in Chapter 9 that yes-no-questions in English have **aux-inversion** (the auxiliary verb precedes the subject) statements have declarative syntax (no aux-inversion), and commands have imperative syntax (sentences with no syntactic subject), as in example (19.19):

(19.19) YES-NO-QUESTION Will breakfast be served on USAir 1557?
STATEMENT I don't care about lunch
COMMAND Show me flights from Milwaukee to Orlando.

It seems from (19.19) that the surface syntax of the input ought to tell us what illocutionary act it is. Alas, as is clear from Abbott and Costello's famous *Who's on First* routine at the beginning of the chapter, things are not so simple. The mapping between surface form and illocutionary act is not obvious or even one-to-one. For example, the following utterance spoken to an ATIS system looks like a

[assert]	C_1:	. . . I need to travel in May.
[info-req,ack]	A_1:	And, what day in May did you want to travel?
[assert, answer]	C_2:	OK uh I need to be there for a meeting that's from the 12th to the 15th.
[info-req,ack]	A_2:	And you're flying into what city?
[assert,answer]	C_3:	Seattle.
[info-req,ack]	A_3:	And what time would you like to leave Pittsburgh?
[check,hold]	C_4:	Uh hmm I don't think there's many options for non-stop.
[accept,ack]	A_4:	Right.
[assert]		There's three non-stops today.
[info-req]	C_5:	What are they?
[assert, open-option]	A_5:	The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.
[accept,ack]	C_6:	OK I'll take the 5ish flight on the night before on the 11th.
[check,ack]	A_6:	On the 11th?
[assert,ack]		OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.
[ack]	C_7:	OK.

Figure 19.21 A potential DAMSL labeling of the conversation fragment in Figure 19.17.

YES-NO-QUESTION meaning something like *Are you capable of giving me a list of. . . ?*:

(19.20) Can you give me a list of the flights from Atlanta to Boston?

In fact, however, this person was not interested in whether the system was *capable* of giving a list; this utterance was actually a polite form of a DIRECTIVE or a REQUEST, meaning something more like *Please give me a list of. . . .* Thus what looks on the surface like a QUESTION can really be a REQUEST.

Similarly, what looks on the surface like a STATEMENT can really be a QUESTION. A very common kind of question, called a CHECK question (Carletta et al., 1997b; Labov and Fanshel, 1977), is used to ask the other participant to confirm something that this other participant has privileged knowledge about. These CHECKs are questions, but they have declarative surface form, as the boldfaced utterance in the following snippet from another travel agent conversation:

A	OPEN-OPTION	I was wanting to make some arrangements for a trip that I'm going to be taking uh to LA uh beginning of the week after next.
B	HOLD	OK uh let me pull up your profile and I'll be right with you here. [pause]
B	CHECK	**And you said you wanted to travel next week?**
A	ACCEPT	Uh yes.

Utterances which use a surface statement to ask a question, or a surface question to issue a request, are called **indirect speech acts**. How can a surface yes-no-question like *Can you give me a list of the flights from Atlanta to Boston?* be mapped into the correct illocutionary act REQUEST?

INDIRECT SPEECH ACTS

Dialogue act interpretation can be approached like any other supervised classification task, by treating the dialog act labels as hidden classes to be estimated by a statistical algorithm trained on a corpus of dialogues that is hand-labeled with dialogue acts for each utterance. Many different features, such as lexical, collocational, syntactic, prosodic, or conversational-structure cues, have been proposed for dialogue act interpretation. Which cues are used depends on the individual system. Many systems rely on the fact that individual dialogue acts often have what Goodwin (1996) called a **microgrammar**; specific lexical, collocation, and prosodic features which are characteristic of them. These systems also rely on conversational structure. The dialogue-act interpretation system of Jurafsky et al. (1997), for example, relies on 3 sources of information:

MICROGRAMMAR

1. **Words and Collocations:** *Please* or *would you* is a good cue for a REQUEST, *are you* for YES-NO-QUESTIONs.
2. **Prosody:** Rising pitch is a good cue for a YES-NO-QUESTION. Loudness or stress can help distinguish the *yeah* that is an AGREEMENT from the *yeah* that is a BACKCHANNEL.
3. **Conversational Structure:** A *yeah* which follows a proposal is probably an AGREEMENT; a *yeah* which follows an INFORM is probably a BACKCHANNEL.

One popular statistical model for integrating these cues into a dialogue act classifier is to use the HMM structure that we've seen for concept labeling in Figure 19.5 (Woszczyna and Waibel, 1994; Reithinger et al., 1996; Kita et al., 1996; Taylor et al., 1998; Stolcke et al., 2000).

NEW FIGURE HERE.

Figure 19.22 HMM approach to dialogue act recognition.

In the HMM approach, given all available evidence E about a conversation,

the goal is to find the dialogue act sequence $D = \{d_1, d_2 \ldots, d_N\}$ that has the highest posterior probability $P(D|E)$ given that evidence (as usual here we use capital letters to mean sequences). Applying Bayes' Rule we get

$$\begin{aligned} D^* &= \operatorname*{argmax}_D P(D|E) \\ &= \operatorname*{argmax}_D \frac{P(D)P(E|D)}{P(E)} \\ &= \operatorname*{argmax}_D P(D)P(E|D) \end{aligned} \tag{19.21}$$

Our survey above suggests that useful types of evidence include (at least) prosody and lexical/collocation information. If we make the simplifying (but of course incorrect) assumption that the prosody and the words are independent, we can estimate the evidence likelihood for a sequence of dialogue acts D as in (19.22):

$$P(E|D) = P(F|D)P(W|D) \tag{19.22}$$

$$D^* = \operatorname*{argmax}_D P(D)P(F|D)P(W|D) \tag{19.23}$$

The resulting equation (19.23) thus has three components, one for each of the kinds of cues discussed above. Let's briefly discuss each of these three components.

The prior probability of a sequence of dialogue acts $P(D)$ acts as a model of conversational structure. Drawing on the idea of adjacency pairs (Schegloff, 1968; Sacks et al., 1974) introduced above, we can make the simplifying assumption that conversational structure is modeled as a Markov sequence of dialogue acts (Nagata and Morimoto, 1994; Suhm and Waibel, 1994; Warnke et al., 1997; Chu-Carroll, 1998; Stolcke et al., 1998; Taylor et al., 1998):

$$P(D) = \prod_{i=2}^{M} P(d_i|d_{i-1}...d_{i-M+1}) \tag{19.24}$$

Woszczyna and Waibel (1994) give the dialogue HMM shown in Figure 19.23 for a Verbmobil-like appointment scheduling task.

The lexical component of the HMM likelihood, designed to capture the microgrammar structure of each dialogue act, is generally modeled by training a separate word-N-gram grammar for each dialogue act, just as we saw with the concept HMM (see e.g., Nagata and Morimoto, 1994; Suhm and Waibel, 1994; Mast et al., 1996; Jurafsky et al., 1997; Warnke et al., 1997; Reithinger and Klesen, 1997; Taylor et al., 1998):

$$P(W|D) = \prod_{i=2}^{N} P(w_i|w_{i-1}...w_{i-N+1}, d_i) \tag{19.25}$$

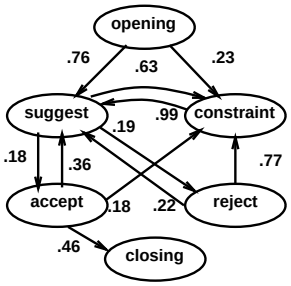

Figure 19.23 A dialogue act HMM (after Woszczyna and Waibel (1994))

Prosodic models of dialogue act microgrammar rely on accents, boundaries, or their acoustic correlates like F0, duration, and energy. For example the pitch rise at the end of YES-NO-QUESTIONS can be a useful cue for augmenting lexical cues (Sag and Liberman, 1975; Pierrehumbert, 1980; Waibel, 1988; Daly and Zue, 1992; Kompe et al., 1993; Taylor et al., 1998). Declarative utterances (like STATEMENTS) have **final lowering**: a drop in F0 at the end of the utterance (Pierrehumbert, 1980). FINAL LOWERING

Shriberg et al. (1998), trained CART-style decision trees on simple acoustically-based prosodic features such as the slope of F0 at the end of the utterance, the average energy at different places in the utterance, and various duration measures, normalized in various ways. They found that these features were useful, for example, in distinguishing the four dialogue acts STATEMENT (S), YES-NO QUESTION (QY), DECLARATIVE-QUESTIONS like CHECKS (QD) and WH-QUESTIONS (QW). Figure 19.24 shows the decision tree which gives the posterior probability $P(d|F)$ of a dialogue act d type given sequence of acoustic features F. Note that the difference between S and QY toward the right of the tree is based on the feature `norm_f0_diff` (normalized difference between mean F0 of end and penultimate regions), while the difference between WQ and QD at the bottom left is based on `utt_grad`, which measures F0 slope across the whole utterance.

Decision trees produce a posterior probability $P(d|f)$, and equation (19.23) requires a likelihood $P(F|d)$. Therefore we need to massage the output of the decision tree by Bayesian inversion (dividing by the prior $P(d_i)$ to turn it into a likelihood. If we make the simplifying assumption that the prosodic decisions for each sentence are independent of other sentences, we arrive at the following final equation for HMM tagging of dialogue acts:

$$D^* = \operatorname*{argmax}_D P(D)P(F|D)P(W|D)$$

$$= \prod_{i=2}^{M} P(d_i|d_{i-1}...d_{i-M+1}) \prod_{i=1}^{N} \frac{P(d_i|F)}{P(d_i)} \prod_{i=2}^{N} P(w_i|w_{i-1}...w_{i-N+1}, d_i) \quad (19.26)$$

Standard HMM decoding techniques (like Viterbi) can then be used to search for this most-probable sequence of dialogue acts given the sequence of input utterances.

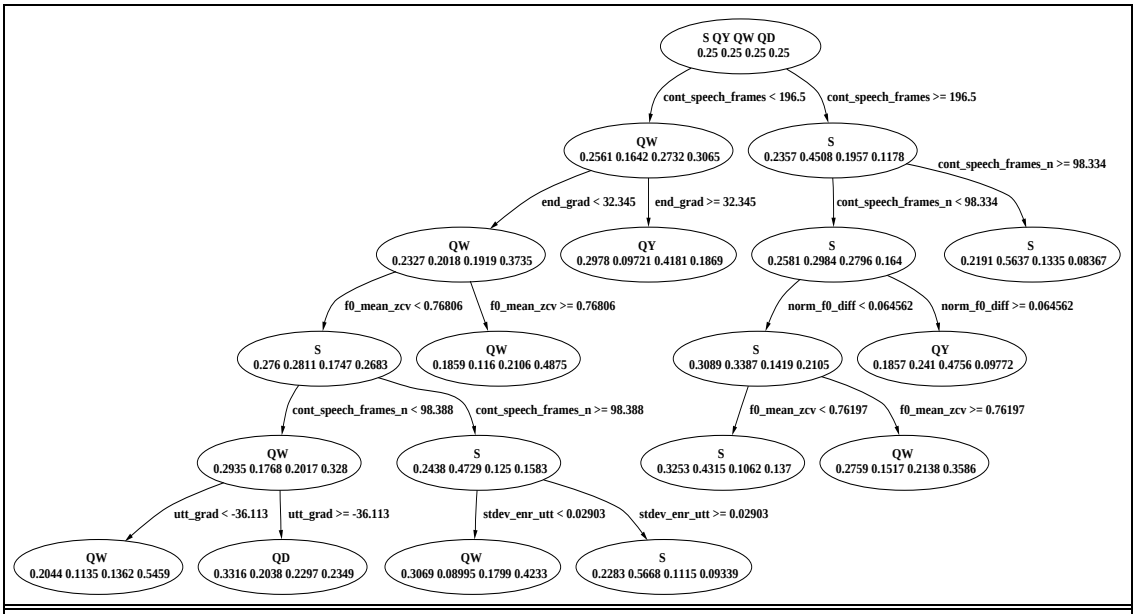

Figure 19.24 Decision tree for the classification of STATEMENT (S), YES-NO QUESTIONS (QY), WH-QUESTIONS (QW) and DECLARATIVE QUESTIONS (QD), after Shriberg et al. (1998). Each node in the tree shows four probabilities, one for each of the four dialogue acts in the order S, QY, QW, QD; the most likely of the four is shown as the label for the node.

The HMM method is only one way of solving the problem of dialogue act identification. Various methods have been applied, such as Transformation-Based Learning (Samuel et al., 1998). In Section 19.9 we will introduce the BDI model, which incorporates a much more complex model of speech act/dialogue act interpretation based on plan-inference.

19.7 Advanced Issue: Computational Processing of Human-Human Dialog

In addition to work on building conversational agents, computational dialogue work also focuses on human-human dialogue. Human-human dialogue is inter-

esting for tasks like automatically transcribing or summarizing business meeting, close-captioning TV shows like news interviews, or building personal telephone assistants that can take notes on telephone conversations.

Human-human speech has a number of differences from human-machine speech. Human-human dialogues generally are more complex than human-machine dialogues, with more syntactic variation, more pronouns with more distant antecedents (?; Guindon, 1988; ?), and more cases of massive phonetic reduction.

One key task in human-human dialogue that does not occur in human-machine dialogue has to do with turn and utterance segmentation. In the rest of this section, we have chosen this one research area as a archetypal task for human-human dialogue.

Turns and Utterances

TURN-TAKING

Dialogue is characterized by **turn-taking**; Speaker A says something, then speaker B, then speaker A, and so on. How do speakers know when is the proper time to contribute their turn? Consider the timing of the utterances in normal human conversations. First, human dialogue have relatively little noticeable overlap. That is, the beginning of each speaker's turn follows the end of the previous speaker's turn.The actual amount of overlapped speech in American English conversation seems to be quite small; Levinson (1983) suggests the amount is less than 5% in general, and probably less for certain kinds of dialogue like the task-oriented dialogue in Figure 19.17. If speakers aren't overlapping, perhaps they are waiting a while after the other speaker? This is also very rare. The amount of time between turns is quite small, generally less than a few hundred milliseconds even in multi-party discourse. In fact, it may take more than this few hundred milliseconds for the next speaker to plan the motor routines for producing their utterance, which means that speakers begin motor planning for their next utterance before the previous speaker has finished. For this to be possible, natural conversation must be set up in such a way that (most of the time) people can quickly figure out **who** should talk next, and exactly **when** they should talk. This kind of turn-taking behavior is generally studied in the field of **Conversation Analysis** (**CA**). In a key conversation-analytic paper, Sacks et al. (1974) argued that turn-taking behavior, at least in American English, is governed by a set of turn-taking rules. These rules apply at a **transition-relevance place**, or **TRP**; places where the structure of the language allows speaker shift to occur. Here is a simplified version of the turn-taking rules, grouped into a single three-part rule; see Sacks et al. (1974) for the complete rules:

CONVERSATION ANALYSIS

(19.27) **Turn-taking Rule.** At each TRP of each turn:

a. If during this turn the current speaker has selected A as the next speaker then A must speak next.
b. If the current speaker does not select the next speaker, any other speaker may take the next turn.
c. If no one else takes the next turn, the current speaker may take the next turn.

There are a number of important implications of rule (19.27) for dialogue modeling. First, subrule (19.27a) implies that there are some utterances by which the speaker specifically selects who the next speaker will be. The most obvious of these are questions, in which the speaker selects another speaker to answer the question. Two-part structures like QUESTION-ANSWER are called **adjacency pairs** (Schegloff, 1968); other adjacency pairs include GREETING followed by GREETING, COMPLIMENT followed by DOWNPLAYER, REQUEST followed by GRANT. We already saw in the previous section that these pairs of dialogue acts and the dialogue expectations they set up play an important role in dialogue modeling.

ADJACENCY PAIRS

Subrule (19.27a) also has an implication for the interpretation of silence. While silence can occur after any turn, silence which follows the first part of an adjacency pair-part is **significant silence**. For example Levinson (1983) notes the following example from Atkinson and Drew (1979); pause lengths are marked in parentheses (in seconds):

SIGNIFICANT SILENCE

(19.28) A: Is there something bothering you or not?
(1.0)
A: Yes or no?
(1.5)
A: Eh?
B: No.

Since A has just asked B a question, the silence is interpreted as a refusal to respond, or perhaps a **dispreferred** response (a response, like saying "no" to a request, which is stigmatized). By contrast, silence in other places, for example a lapse after a speaker finishes a turn, is not generally interpretable in this way. These facts are relevant for user interface design in spoken dialogue systems; users are disturbed by the pauses in dialogue systems caused by slow speech recognizers (Yankelovich et al., 1995).

DISPREFERRED

Automatic segmentation of utterances in human-human conversation

Another implication of (19.27) is that transitions between speakers don't occur just anywhere; the **transition-relevance places** where they tend to occur are generally at **utterance** boundaries. This brings us to the next difference between spoken di-

UTTERANCE

alogue and textual monologue (of course dialogue can be written and monologue spoken; but most current applications of dialogue involve speech): the spoken **utterance** versus the written **sentence**. Recall from Chapter 9 that utterances differ from written sentences in a number of ways. They tend to be shorter, are more likely to be single clauses, the subjects are usually pronouns rather than full lexical noun phrases, and they include filled pauses, repairs, and restarts.

One very important difference not discussed in Chapter 9 is that while written sentences and paragraphs are relatively easy to automatically segment from each other, utterances and turns are quite complex to segment. Utterance boundary detection is important since many computational dialogue models are based on extracting an utterance as a primitive unit. The segmentation problem is difficult because a single utterance may be spread over several turns, or a single turn may include several utterances. For example in the following fragment of a dialogue between a travel agent and a client, the agent's utterance stretches over three turns:

(19.29)

A: Yeah yeah the um let me see here we've got you on American flight nine thirty eight
C: Yep.
A: leaving on the twentieth of June out of Orange County John Wayne Airport at seven thirty p.m.
C: Seven thirty.
A: and into uh San Francisco at eight fifty seven.

By contrast, the example below has three utterances in one turn:

(19.30)

A: Three two three and seven five one. OK and then does he know there is a nonstop that goes from Dulles to San Francisco? Instead of connection through St. Louis.

Algorithms for utterance segmentation are based on many boundary **cues** such as:

- **cue words:** Cue (or "clue") words like *well*, *and*, *so*, etc., tend to occur at the beginnings and ends of utterances (Reichman, 1985; Hirschberg and Litman, 1993). CUE WORDS
- ***N*-gram word or POS sequences:** Specific word or POS sequences often indicate boundaries. *N*-gram grammars can be trained on a training set labeled with special utterance-boundary tags, and then a decoder can find the most likely utterance boundaries in an unlabeled test set (Mast et al., 1996; Meteer and Iyer, 1996; Stolcke and Shriberg, 1996; Heeman and Allen, 1999).
- **prosody:** Prosodic features like pitch, accent, phrase-final lengthening and pause duration play a role in utterance/turn segmentation, as discussed in

INTONATION PHRASE

Chapter 4, although the relationship between utterances and prosodic units like the **intonation unit** (Bois et al., 1983) or **intonation phrase** (Pierrehumbert, 1980; Beckman and Pierrehumbert, 1986) is complicated (Ladd, 1996; Ford and Thompson, 1996; Ford et al., 1996, inter alia) .

19.8 ADVANCED: MARKOV DECISION PROCESSES

Earlier we discussed how dialogue systems could change confirmation strategies based on context. For example if the ASR or NLU confidence is low, we might choose to do explicit confirmation. If confidence is high, we might chose implicit confirmation, or even decide not to confirm at all. Using a dynamic strategy lets us choose the action which maximizes dialogue success, while minimizing costs. This idea of changing the actions of a dialogue system based on optimizing some kinds of rewards or costs is the fundamental intuition behind modeling dialogue as a **Markov decision process**.

MARKOV DECISION PROCESS

MDP

A Markov decision process or **MDP** is characterized by a set of **states** *S* an agent can be in, a set of **actions** *A* the agent can take, and a **reward** r(a,s) that the agent receives for taking an action in a state. Given these factors, we can compute a **policy** π which specifies which action *a* the agent should take when in a given state *s*, so as to receive the best reward. To understand each of these components, let's look at a tutorial example of an MDP implementation taken from (Levin et al., 2000). Their tutorial example is a "Day-and-Month" dialogue system, whose goal is to get correct values of day and month for a two-slot frame via the shortest possible interaction with the user.

In principle, a state of an MDP could include any possible information about the dialogue, such as the complete dialogue history so far. Using such a rich model of state would make the number of possible states extraordinarily large. So a model of state is usually chosen which encodes a much more limited set of information, such as the values of the slots in the current frame, the most recent question asked to the user, the users most recent answer, the ASR confidence, and so on. For the Day-and-Month example let's represent the state of the system as the values of the two slots *day* and *month*. If we assume a special initial state s_i and final state s_f, there are a total of 411 states (366 states with a day and month (counting leap year), 12 states with a month but no day (d=0, m= 1,2,...12), and 31 states with a day but no month (m=0, d=1,2,...31)).

Actions of a MDP dialogue system might include generating particular speech acts, or performing a database query to find out information. For the Day-and-Month example, (Levin et al., 2000) propose the following actions:

- a_d: a question asking for the day
- a_m: a question asking for the month
- $a_d m$: a question asking for both the day and the month
- a_f: a final action submitting the form and terminating the dialogue

Since the goal of the system is to get the correct answer with the shortest interaction, one possible reward function for the system would integrate three terms:

$$R = -(w_i n_i + w_e n_e + w_f n_f) \tag{19.31}$$

The term n_i is the number of interactions with the user, n_e is the number of errors, n_f is the number of slots which are filled (0, 1, or 2), and the ws are weights.

Finally, a dialogue policy π specifies which actions to apply in which state. Consider two possible policies: (1) asking for day and month separately, and (2) asking for them together. These might generate the two dialogues shown in Figure 19.25.

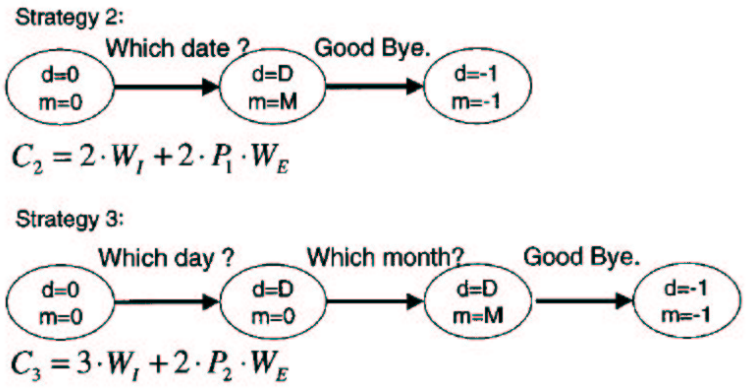

Figure 19.25 LEVIN FIGURE 2 HERE, NEED TO REDRAW WITH JUST 2 POLICIES, IN REVERSE ORDER, after (Levin et al., 2000).PLACEHOLDER FIGURE.

In policy 1, the action specified for the no-date/no-month state is to ask for a day, while the action specified for any of the 31 states where we have a day but not a month is to ask for a month. In policy 2, the action specified for the no-date/no-month state is to ask an open-ended question (*Which date*) to get both a day and a month. The two policies have different advantages; an open prompt can leads to shorter dialogues but is likely to cause more errors, while a directive prompt is slower but less error-prone. Thus the optimal policy depends on the values of the weights w, and also on the error rates of the ASR component. Let's call p_d the probability of the recognizer making an error interpreting a month or

a day value after a directive prompt. The (presumably higher) probability of error interpreting a month or day value after an open prompt we'll call p_o. The reward for the first dialog in Figure 19.25 is thus $-3 \times w_i + 2 \times p_d \times w_e$. The reward for the second dialog in Figure 19.25 is $-2 \times w_i + 2 \times p_d \times w_e$. The directive prompt policy, policy 2, is thus better than policy 1 when the improved error rate justifies the longer interaction, i.e., when $p_o - p_d > \frac{w_i}{2w_e}$.

In the example we've seen so far, there were only two possible actions, and hence only a tiny number of possible policies. In general, the number of possible actions, states, and policies is quite large, and so the problem of finding the optimal policy π^* is much harder.

Markov decision theory together with classical reinforcement learning gives us a way to think about this problem. First, generalizing from Figure 19.25, we can think of any particular dialogue as a trajectory in state space:

$$s_1 \rightarrow_{a1,r1} s_2 \rightarrow_{a2,r2} s_3 \rightarrow_{a3,r3} \cdots \tag{19.32}$$

The best policy π^* is the one with the greatest expected reward over all trajectories. What is the expected reward for a given state sequence? The most common way to assign utilities or rewards to sequences is to use **discounted rewards**. Here we compute the expected cumulative reward Q of a sequence as a discounted sum of the utilities of the individual states:

DISCOUNTED REWARDS

$$Q([s_0, a_0, s_1, a_1, s_2, a_2 \cdots]) = R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots, \tag{19.33}$$

The discount factor γ is a number between 0 and 1. This makes the agent care more about current rewards than future rewards; the more future a reward, the more discounted its value.

Given this model, it is possible to show that the expected cumulative reward $Q(s,a)$ for taking a particular action from a particular state is the following recursive equation called the **Bellman equation**:

BELLMAN EQUATION

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a') \tag{19.34}$$

What the Bellman equation says is that the expected cumulative reward for a given state/action pair is the immediate reward for the current state plus the expected discounted utility of all possible next states s', weighted by the probability of moving to that state s', and assuming once there we take the optimal action a.

Equation (19.34) makes use of two parameters. We need a model of how likely a given state/action pair (s,a) is to lead to a new state s'. And we also need a good estimate of $R(s,a)$. If we had lots of labeled training data, we could simply compute both of these from labeled counts. For example, with labeled dialogues, we could simply count how many times we were in a given state s, and out of that how many times we took action a to get to state s', to estimate $P(s'|s,a)$. Similarly,

if we had a hand-labeled reward for each dialogue, we could build a model of $R(s,a)$.

Given these parameters, it turns out that there is an iterative algorithm for solving the Bellman equation and determining proper Q values, the **value iteration** algorithm (). We won't present this here, but see Chapter 17 of (Russell and Norvig, 2002) for the details of the algorithm as well as further information on Markov Decision Processes.

VALUE ITERATION

How do we get enough labeled training data to set these parameters? This is especially worrisome in any real problem, where the number of states s is extremely large. Two methods have been applied in the past. The first is to carefully hand-tune the states and policies so that there are a very small number of states and policies that need to be set automatically. In this case we can build a dialogue system which explore the state space by generating random conversations. Probabilities can then be set from this corpus of conversations. The second is to build a simulated user. The user interacts with the system millions of times, and the system learns the state transition and reward probabilities from this corpus.

The random conversation approach was taken by (Singh et al., 2002). They used reinforcement learning to make a small set of optimal policy decisions. Their NJFun system learned to choose actions which varied the initiative (system, user, or mixed) and the confirmation strategy (explicit or none). The state of the system was specified by values of 7 features including which slot in the frame is being worked on (1-4), the ASR confidence value (0-5), how many times a current slot question had been asked, whether a restritive or non-restricive grammar was used, and so on. The result of using only 7 features with a small number of attributes resulted in a small state space (62 states). Each state had only 2 possible actions (system versus user initiative when asking questions, explicit versus no confirmation when receiving answers). They ran the system with real users, creating 311 conversations. Each conversation had a very simple binary reward function; 1 if the user completed the task (finding specified museums, theater, winetasting in the New Jersey area), 0 if the user did not. The system successful learned a good dialogue policy (roughly, start with user initiative, then back of to either mixed or system initiative when reasking for an attribute; confirm only at lower confidence values; both initiative and confirmation policies, however, are different for different attributes). They showed that their policy actually was more succesful based on various objective measures than many hand-designed policies reported in the literature.

The simulated user strategy was taken by (Levin et al., 2000), in their MDP model with reinforcement learning in the ATIS task. Their simulated user was a generative stochastic model that given the system's current state and actions, produces a frame-slot representation of a user response. The parameters of the sim-

ulated user were estimated form a corpus of ATIS dialogues. The simulated user was then used to interact with the system for tens of thousands of conversations, leading to an optimal dialogue policy.

FIX: Add 2 pgraphs on POMDP, including Roy et al, Young 2002, and Williams and Young.

19.9 ADVANCED: PLAN-BASED DIALOGUE AGENTS

One of the earliest models of conversational agent behavior, and also one of the most sophisticated, is based on the use of planning techniques from early AI models. The idea is that communication and conversation are just special cases of rational action in the world, and these actions can be planned like any other.

Such plan-based conversation agents are used in building agents to help with problems where planning is already a necessary part of the system. For example, the Rochester TRIPS system (Allen et al., 2001) models a conversational agent that helps with emergency management (planning where and how to supply ambulances or personnel in a simulated emergency situation). Solving such problems (e.g., deciding whether and how to get an ambulance from point A to point B) requires sophisticated models of planning and reasoning. The plan-based approach to dialogue applies these same planning algorithms to conversation as well.

The idea that actions in conversation should be planned just like actions in the real world takes its fundamental intuition from the ideas of speech acts described in Section 19.5. For example, planning can be used to *generate* speech acts. One agent, seeking to find out some information, could use standard planning techniques to come up with the plan of asking the interlocuter to tell the first agent the information. Planning can also be used to *interpret* speech acts, by running the planner 'in reverse'. An agent hearing an utterance can use inference rules to infer what plan the interlocuter might have had to cause them to say what they said.

Using plans to generate and interpret sentences in this way require that the planner have good models of what its own goals and knowledge are, as well as the goals and knowledge of the interlocuter. These planners thus need to model the **beliefs**, **desires**, and **intentions** (BDI) of the interlocuter, and hence plan-based models of dialogue are referred to as **BDI** models. BDI models of dialogue were first introduced by Allen, Cohen, Perrault, and their colleagues and students in a number of influential papers showing how speech acts could be generated (Cohen and Perrault, 1979), and interpreted (Perrault and Allen, 1980; Allen and Perrault, 1980). In a parallel line of research, Grosz and her colleagues and students showed how using similar notions of intention and plans allowed the kind of conversational structure and coherence discussed in Chapter 18 to be applied to dialogue. We will

BDI

explore both these lines of research in this section.

Conversational Implicature

One of the guiding motivations for the BDI paradigm for building plan-based agents is the role that inference plays in conversation. We begin with that motivation, considering the way the interpretation of an utterance relies on more than just the literal meaning of the sentences. Consider the client's response C_2 from the sample conversation in Figure 19.17, repeated here:

A_1: And, what day in May did you want to travel?

C_2: OK uh I need to be there for a meeting that's from the 12th to the 15th.

Notice that the client does not in fact answer the question. The client merely states that he has a meeting at a certain time. The semantics for this sentence produced by a semantic interpreter will simply mention this meeting. What is it that licenses the agent to infer that the client is mentioning this meeting so as to inform the agent of the travel dates?

Now consider another utterance from the sample conversation, this one by the agent:

A_4: ... There's three non-stops today.

Now this statement would still be true if there were seven non-stops today, since if there are seven of something, there are by definition also three. But what the agent means here is that there are three **and not more than three** non-stops today. How is the client to infer that the agent means **only three** non-stops?

These two cases have something in common; in both cases the speaker seems to expect the hearer to draw certain inferences; in other words, the speaker is communicating more information than seems to be present in the uttered words. These kind of examples were pointed out by Grice (1975, 1978) as part of his theory of **conversational implicature**. **Implicature** means a particular class of licensed inferences. Grice proposed that what enables hearers to draw these inferences is that conversation is guided by a set of **maxims**, general heuristics which play a guiding role in the interpretation of conversational utterances. He proposed the following four maxims:

IMPLICATURE

MAXIMS

- **Maxim of Quantity:** Be exactly as informative as is required: QUANTITY
  1. Make your contribution as informative as is required (for the current purposes of the exchange).
  2. Do not make your contribution more informative than is required.
- **Maxim of Quality:** Try to make your contribution one that is true: QUALITY
  1. Do not say what you believe to be false.

2. Do not say that for which you lack adequate evidence.

RELEVANCE
- **Maxim of Relevance:** Be relevant.

MANNER
- **Maxim of Manner:** Be perspicuous:
  1. Avoid obscurity of expression.
  2. Avoid ambiguity.
  3. Be brief (avoid unnecessary prolixity).
  4. Be orderly.

It is the Maxim of Quantity (specifically Quantity 1) that allows the hearer to know that *three non-stops* did not mean *seven non-stops*. This is because the hearer assumes the speaker is following the maxims, and thus if the speaker meant seven non-stops she would have said seven non-stops ("as informative as is required"). The Maxim of Relevance is what allows the agent to know that the client wants to travel by the 12th. The agent assumes the client is following the maxims, and hence would only have mentioned the meeting if it was relevant at this point in the dialogue. The most natural inference that would make the meeting relevant is the inference that the client meant the agent to understand that his departure time was before the meeting time.

Plan-Inferential Interpretation and Production

The insight of the Gricean approach to comprehension summarized in the previous section is that in order to understand, the hearer must make inferences about the speaker's knowledge and intention. This idea underlies the use of the planning paradigm in conversational agents. In this section we sketch the BDI model, exploring how a plan-based agent might replace the human travel agent in the conversational fragment discussed above. We'll look at one example of plan-based comprehension and one (simpler) example of plan-based production.

First let's consider how a plan-based agent could act as the human travel agent to understand sentence C_2 in the dialogue repeated below:

C_1: I need to travel in May.

A_1: And, what day in May did you want to travel?

C_2: OK uh I need to be there for a meeting that's from the 12th to the 15th.

As the previous section discussed, the Gricean principle of Relevance can be used to infer that the client's meeting is relevant to the flight booking. The system may know that one precondition for having a meeting (at least before web conferencing) is being at the place where the meeting is in. One way of being at a place is flying there, and booking a flight is a precondition for flying there. The

system can follow this chain of inference, abducing that user wants to fly on a date before the 12th.

Next, consider how our plan-based agent could act as the human travel agent to produce sentence A_1 in the dialogue above. In a plan-based model of this interaction, the planning agent would reason that in order to help a client book a flight it must know enough information about the flight to book it. It reasons that knowing the month (May) is insufficient information to specify a departure or return date. The simplest way to find out the needed date information is simply to ask the client.

Both of these cases, planning for understanding and planning for generation, can be modeled in the BDI framework. In the rest of this section, we'll flesh out the sketchy outlines above. We'll begin by summarizing Perrault and Allen's formal definitions of belief and desire in the predicate calculus. We'll represent "*S* believes the proposition *P*" as the two-place predicate $B(S,P)$. Reasoning about belief is done with a number of axiom schemas inspired by Hintikka (1969) (such as $B(A,P) \wedge B(A,Q) \Rightarrow B(A,P \wedge Q)$; see Perrault and Allen (1980) for details). Knowledge is defined as "true belief"; *S knows that P* will be represented as $KNOW(S,P)$, defined as follows:

$$\text{KNOW}(S,P) \equiv P \wedge B(S,P)$$

The theory of desire relies on the predicate WANT. If an agent *S* wants *P* to be true, we say $WANT(S,P)$, or $W(S,P)$ for short. *P* can be a state or the execution of some action. Thus if ACT is the name of an action, $W(S,\text{ACT}(H))$ means that *S* wants *H* to do ACT. The logic of WANT relies on its own set of axiom schemas just like the logic of belief.

The BDI models also require an axiomatization of actions and planning; the simplest of these is based on a set of **action schema**s based on the simple AI planning model STRIPS (Fikes and Nilsson, 1971). Each action schema has a set of parameters with *constraints* about the type of each variable, and three parts: ACTION SCHEMA

- *Preconditions:* Conditions that must already be true in order to successfully perform the action.
- *Effects:* Conditions that become true as a result of successfully performing the action.
- *Body:* A set of partially ordered goal states that must be achieved in performing the action.

In the travel domain, for example, the action of agent *A* booking flight $F1$ for client *C* might have the following simplified definition:

BOOK-FLIGHT(A,C,F):

Constraints: Agent(A) ∧ Flight(F) ∧ Client(C)
Precondition: Know(A,departure-date(F)) ∧ Know(A,departure-time(F)) ∧ Know(A,origin-city(F)) ∧ Know(A,destination-city(F)) ∧ Know(A,flight-type(F)) ∧ Has-Seats(F) ∧ W(C,(BOOK(A,C,F))) ∧ ...

Effect: Flight-Booked(A,C,F)
Body: Make-Reservation(A,F,C)

Cohen and Perrault (1979) and Perrault and Allen (1980) use this kind of action specification for speech acts. For example here is Perrault and Allen's definition for two speech acts. INFORM is the speech act of informing the hearer of some proposition (the Austin/Searle *Assertive*, or DAMSL STATEMENT). The definition of INFORM is based on Grice's (1957) idea that a speaker informs the hearer of something merely by causing the hearer to believe that the speaker wants them to know something:

INFORM(S,H,P):
Constraints: Speaker(S) ∧ Hearer(H) ∧ Proposition(P)
Precondition: Know(S,P) ∧ W(S, INFORM(S, H, P))
Effect: Know(H,P)
Body: B(H,W(S,Know(H,P)))

REQUEST is the directive speech act for requesting the hearer to perform some action:

REQUEST(S,H,ACT):
Constraints: Speaker(S) ∧ Hearer(H) ∧ ACT(A) ∧ H is agent of ACT
Precondition: W(S,ACT(H))
Effect: W(H,ACT(H))
Body: B(H,W(S,ACT(H)))

Let's now see how a plan-based dialogue system might try to understand the sentence:

C_2: OK uh I need to be there for a meeting that's from the 12th to the 15th.

We'll assume the system has the BOOK-FLIGHT plan mentioned above. In addition, we'll need knowledge about meetings and getting to them, in the form of the MEETING, FLY-TO, and TAKE-FLIGHT plans, sketched broadly below:

MEETING(P,L,T1,T2):

Constraints: Person(P) ∧ Location (L) ∧ Time (T1) ∧ Time (T2) ∧ Time (TA)
Precondition: At (P, L, TA)
Before (TA, T1)
Body: ...

FLY-TO(P, L, T):
Constraints: Person(P) ∧ Location (L) ∧ Time (T)
Effect: At (P, L, T)
Body: TAKE-FLIGHT(P, L, T)

TAKE-FLIGHT(P, L, T):
Constraints: Person(P) ∧ Location (L) ∧ Time (T) ∧ Flight (F) ∧ Agent (A)
Precondition: BOOK-FLIGHT (A, P, F)
Destination-Time(F) = T
Destination-Location(F) = L
Body: ...

Now let's assume that an NLU module returns a semantics for the client's utterance which (among other things) includes the following semantic content:

MEETING (P, ?L, T1, T2) Constraints: P = Client ∧ T1 = May 12 ∧ T2 = May 15

Our plan-based system now has two plans established, one MEETING plan from this utterance, and one BOOK-FLIGHT plan from the previous utterance. The system implicitly uses the Gricean Relevance intuition to try to connect them. Since BOOK-FLIGHT is a precondition for TAKE-FLIGHT, the system may hypothesize (infer) that the user is planning a TAKE-FLIGHT. Since TAKE-FLIGHT is in the body of FLY-TO, the system further infers a FLY-TO plan. Finally, since the effect of FLY-TO is a precondition of the MEETING, the system can unify each of the people, locations, and times of all of these plans. The result will be that the system knows that the client wants to arrive at the destination before May 12th.

Let's turn to the details of our second example; how our plan-based agent could act as the human travel agent to produce sentence A_1, repeated here:

C_1: I need to travel in May.
A_1: And, what day in May did you want to travel?

How does a plan-based agent know to ask question A_1? This knowledge comes from the BOOK-FLIGHT plan, whose preconditions were that the agent know a variety of flight parameters including the departure date and time, origin and destination cities, and so forth. Utterance C_1 contains the origin city and partial information about the departure date; the agent has to request the rest. A plan-based agent would use an action schema like REQUEST-INFO to represent a plan for asking information questions (simplified from Cohen and Perrault (1979)):

REQUEST-INFO(A,C,I):

Constraints:	Agent(A) $\wedge$ Client(C)
Precondition:	Know(C,I)
Effect:	Know(A,I)
Body:	B(C,W(A,Know(A,I)))

Because the effects of REQUEST-INFO match each precondition of BOOK-FLIGHT, the agent can use REQUEST-INFO to achieve the missing information.

This overview of the BDI model was of necessity very brief. The interested reader should consult the literature suggested at the end of the chapter.

Dialogue Structure and Coherence

Section **??** described an approach to determining coherence based on a set of coherence relations. In order to determine that a coherence relation holds, the system must reason about the constraints that the relation imposes on the **information** in the utterances. We will call this view the *informational* approach to coherence. Historically, the informational approach has been applied predominantly to monologues.

The BDI approach to utterance interpretation gives rise to another view of coherence, which we will call the **intentional** approach. According to this approach, utterances are understood as actions, requiring that the hearer infer the plan-based speaker intentions underlying them in establishing coherence. In contrast to the informational approach, the intentional approach has been applied predominantly to dialogue.

The intentional approach we describe here is due to Grosz and Sidner (1986), who argue that a discourse can be represented as a composite of three interacting components: a **linguistic structure**, an **intentional structure**, and an **attentional state**. The linguistic structure contains the utterances in the discourse, divided into a hierarchical structure of discourse segments. (Recall the description of discourse segments in Chapter 18.) The attentional state is a dynamically-changing model of the objects, properties, and relations that are salient at each point in the discourse. This aligns closely with the notion of a discourse model introduced in the previous chapter. Centering (see Chapter 18) is considered to be a theory of attentional state in this approach.

LINGUISTIC STRUCTURE

INTENTIONAL STRUCTURE

ATTENTIONAL STATE

We will concentrate here on the third component of the approach, the intentional structure, which is based on the BDI model of interpretation. The fundamental idea is that a discourse has associated with it an underlying purpose that is held by the person who initiates it, called the **discourse purpose** (DP). Likewise, each discourse segment within the discourse has a corresponding purpose, called a **discourse segment purpose** (DSP). Each DSP has a role in achieving the DP of

DISCOURSE PURPOSE

DISCOURSE SEGMENT PURPOSE

the discourse in which its corresponding discourse segment appears. Listed below are some possible DPs/DSPs that Grosz and Sidner give.

1. Intend that some agent intend to perform some physical task.
2. Intend that some agent believe some fact.
3. Intend that some agent believe that one fact supports another.
4. Intend that some agent intend to identify an object (existing physical object, imaginary object, plan, event, event sequence).
5. Intend that some agent know some property of an object.

As opposed to the larger sets of coherence relations used in informational accounts of coherence, Grosz and Sidner propose only two such relations: **dominance** and **satisfaction-precedence**. DSP_1 dominates DSP_2 if satisfying DSP_2 is intended to provide part of the satisfaction of DSP_1. DSP_1 satisfaction-precedes DSP_2 if DSP_1 must be satisfied before DSP_2.

As an example, let's consider the dialogue between a client (C) and a travel agent (A) that we saw earlier, repeated here in Figure 19.26.

C_1:	I need to travel in May.
A_1:	And, what day in May did you want to travel?
C_2:	OK uh I need to be there for a meeting that's from the 12th to the 15th.
A_2:	And you're flying into what city?
C_3:	Seattle.
A_3:	And what time would you like to leave Pittsburgh?
C_4:	Uh hmm I don't think there's many options for non-stop.
A_4:	Right. There's three non-stops today.
C_5:	What are they?
A_5:	The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.
C_6:	OK I'll take the 5ish flight on the night before on the 11th.
A_6:	On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.
C_7:	OK.

Figure 19.26 A fragment from a telephone conversation between a client (C) and a travel agent (A) (repeated from Figure 19.17).

Collaboratively, the caller and agent successfully identify a flight that suits the caller's needs. Achieving this joint goal required that a top-level discourse intention be satisfied, listed as I1 below, in addition to several intermediate intentions that contributed to the satisfaction of I1, listed as I2-I5:

I1: (Intend C (Intend A (A find a flight for C)))

I2: (Intend A (Intend C (Tell C A departure date)))

I3: (Intend A (Intend C (Tell C A destination city)))

I4: (Intend A (Intend C (Tell C A departure time)))

I5: (Intend C (Intend A (A find a nonstop flight for C)))

Intentions I2–I5 are all subordinate to intention I1, as they were all adopted to meet preconditions for achieving intention I1. This is reflected in the dominance relationships below:

I1 dominates I2

I1 dominates I3

I1 dominates I4

I1 dominates I5

Furthermore, intentions I2 and I3 needed to be satisfied before intention I5, since the agent needed to know the departure date and destination city in order to start listing nonstop flights. This is reflected in the satisfaction-precedence relationships below:

I2 satisfaction-precedes I5

I3 satisfaction-precedes I5

The dominance relations give rise to the discourse structure depicted in Figure 19.27. Each discourse segment is numbered in correspondence with the intention number that serves as its DP/DSP.

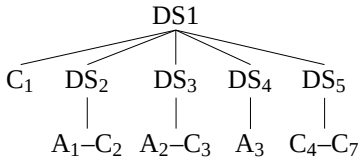

Figure 19.27 Discourse Structure of the Flight Reservation Dialogue

On what basis does this set of intentions and relationships between them give rise to a coherent discourse? It is their role in the overall *plan* that the caller is inferred to have. We assume that the caller and agent have the plan BOOK-FLIGHT described on page 54. This plan requires that the agent know the departure time and date and so on. As we discussed above, the agent can use the REQUEST-INFO action scheme from page 56 to ask the user for this information.

Discourse segments DS2 and DS3 are cases in which performing REQUEST-INFO succeeds for identifying the values of the departure date and destination

city parameters respectively. Segment DS4 is also a request for a parameter value (departure time), but is unsuccessful in that the caller takes the initiative instead, by (implicitly) asking about nonstop flights. Segment DS5 leads to the satisfaction of the top-level DP from the caller's selection of a nonstop flight from a short list that the agent produced.

Subsidiary discourse segments like DS2 and DS3 are also called **subdialogues**. The type of subdialogues that DS2 and DS3 instantiate are generally called **knowledge precondition** subdialogues (Lochbaum et al., 1990; Lochbaum, 1998), since they are initiated by the agent to help satisfy preconditions of a higher-level goal (in this case addressing the client's request for travel in May). They are also called **information-sharing subdialogues** (Chu-Carroll and Carberry, 1998).

SUBDIALOGUES

INFORMATION-SHARING SUBDIALOGUES

Determining Intentional Structure Algorithms for inferring intentional structure in dialogue (and spoken monologue) work similarly to algorithms for inferring dialogue acts. Many algorithms apply variants of the BDI model (e.g., Litman, 1985; Grosz and Sidner, 1986; Litman and Allen, 1987; Carberry, 1990; Passonneau and Litman, 1993; Chu-Carroll and Carberry, 1998). Others rely on similar cues to those described for utterance- and turn-segmentation on page 45, including cue words and phrases (Reichman, 1985; Grosz and Sidner, 1986; Hirschberg and Litman, 1993), prosody (Grosz and Hirschberg, 1992; Hirschberg and Pierrehumbert, 1986; Hirschberg and Nakatani, 1996), and other cues. For example Pierrehumbert and Hirschberg (1990) argue that intonational events like certain **boundary tones** might be used to suggest a dominance relation between two intonational phrases.

Informational vs. Intentional Coherence As we just saw, the key to intentional coherence lies in the ability of the dialogue participants to recognize each other's intentions and how they fit into the plans they have. On the other hand, as we saw in the previous chapter, informational coherence lies in the ability to establish certain kinds of content-bearing relationships between utterances. So one might ask what the relationship between these are: does one obviate the need for the other, or do we need both?

Moore and Pollack (1992), among others, have argued that in fact both levels of analysis must co-exist. Let us assume that after our agent and caller have identified a flight, the agent makes the statement in passage (19.35).

(19.35) You'll want to book your reservations before the end of the day.
Proposition 143 goes into effect tomorrow.

This passage can be analyzed either from the intentional or informational perspective. Intentionally, the agent intends to convince the caller to book her reservation before the end of the day. One way to accomplish this is to provide motivation

for this action, which is the role served by uttering the second sentence. Informationally, the two sentences satisfy the Explanation relation described in the last chapter, since the second sentence provides a cause for the effect of wanting to book the reservations before the end of the day.

Depending on the knowledge of the caller, recognition at the informational level might lead to recognition of the speaker's plan, or vice versa. Say, for instance, that the caller knows that Proposition 143 imposes a new tax on airline tickets, but did not know the intentions of the agent in uttering the second sentence. From the knowledge that a way to motivate an action is to provide a cause that has that action as an effect, the caller can surmise that the agent is trying to motivate the action described in the first sentence. Alternatively, the caller might have surmised this intention from the discourse scenario, but have no idea what Proposition 143 is about. Again, knowing the relationship between establishing a cause-effect relationship and motivating something, the caller might be led to assume an Explanation relationship, which would require that she infers that the proposition is somehow bad for airline ticket buyers (e.g., a tax). Thus, at least in some cases, both levels of analysis appear to be required.

Dialogue Management in a Plan-based Conversational Agent

The more complex representational and reasoning components of the BDI architecture have implications also for dialogue management. Figure 19.28 shows the dialogue manager algorithm for the TRAINS-93 system Allen et al. (1995), Traum and Allen (1994). The TRAINS system is a conversational agent that assists a user in managing a railway transportation system in a microworld. For example, the user and the system might collaborate in planning to move a boxcar of oranges from one city to another. The TRAINS dialogue manager maintains the flow of conversation and addresses the conversational goals (such as coming up with an operational plan for achieving the domain goal of successfully moving oranges). To do this, the manager must model the state of the dialogue, its own intentions, and the user's requests, goals, and beliefs. The manager uses a conversation act interpreter to semantically analyze the user's utterances, a domain planner and executer to solve the actual transportation domain problems, and a generator to generate sentences to the user.

The algorithm keeps a queue of conversation acts it needs to generate. Acts are added to the queue based on **grounding**, **dialogue obligations**, or the agent's **goals**. Recall from Section 19.4 that utterances can be grounded via acknowledgement (*uh-huh*, *ok*), demonstration/display (repeating back), or making a relevant next contribution. Obligations are used in the TRAINS system to enable the system to correctly produce the second-pair part of an adjacency pair. That is, when a

DIALOGUE_MANAGER

```
     while conversation is not finished
       if user has completed a turn
       then interpret user's utterance
       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 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
```

Figure 19.28 A dialogue manager, modified from Traum and Allen (1994).

user REQUESTs something of the system (e.g., REQUEST(Give(List))) an obligation is created to address the REQUEST either by accepting it, and then performing it (giving the list) or by rejecting it. As for goal, for the travel agent domain, the dialogue manager's goal might be to find out the client's travel goal and then create an appropriate plan.

Let's pretend that the human travel agent for the conversation in Figure 19.26 was a system and explore what the state of a TRAINS-style dialogue manager would have to be to act appropriately. Consider the state of the dialogue manager after the first utterances in our sample conversation:

C_1: I want to go to Pittsburgh in May.

Here the client/user has just finished a turn with an INFORM speech act. The system has the discourse goal of finding out the user's travel goal (e.g., "Wanting to go to Pittsburgh on may 15 and returning ..."), and creating a travel plan to accomplish that goal. The following table shows the system state: obligations, intended speech acts to be passed to the generator, the user's speech acts that still need to be acknowledged, discourse goals, and turn holder:

Discourse obligations:	NONE
Turn holder:	system
Intended speech acts:	NONE
Unacknowledged speech acts:	INFORM-1
Discourse goals:	get-travel-goal, create-travel-plan

After the utterance, the dialogue manager decides to add two conversation acts to the queue; first, to acknowledge the user's INFORM act (via "address grounding situation"), and second, to ask the next question of the user (via "address goals"). This reasoning would be worked out by the system's STRIPS-style planner as described on page 54; given the goal *get-travel-goal*, the REQUEST-INFO action schema tells the system that asking the user something is one way of finding it out. The result of adding these two conversation acts is:

Intended speech acts: REQUEST-INFORM-1, ACKNOWLEDGE-1

These would be combined by a very clever generator into the single utterance:

A_2: And, what day in May did you want to travel?

Note that grounding is achieved by the discourse marker (*and*) and by repeating back the month name *May*. The request for information is achieved via the wh-question.

Let's skip ahead to the client's utterance C_4, an indirect request asking the agent to check on non-stop flights.

A_3: And what time would you like to leave Pittsburgh?
C_4: Uh hmm I don't think there's many options for non-stop.

Let's assume that our dialogue act interpreter correctly interprets C_4 as REQUEST-INFORM-3. The state of the agent after client utterance C_4 is then:

Discourse obligations:	address(REQUEST-INFORM-3)
Turn holder:	system
Intended speech acts:	NONE
Unacknowledged speech acts:	REQUEST-INFORM-3
Discourse goals:	get-travel-goal, create-travel-plan

The dialogue manager will first address the discourse obligation of responding to the user's request by calling the planner to find out how many non-stop flights there are. The system must now answer the question and also ground the user's utterance. For a direct request, the response is sufficient grounding. For an indirect request, an explicit acknowledgement is an option; since the indirect request was in the form of a *negative* check question, the form of acknowledgement will be *right* (*no* would have also been appropriate for acknowledging a negative). These two acts will then be pulled off the queue and passed to the generator:

A_4: Right. There's three non-stops today.

Although this TRIPS dialogue manager successfully deals with grounding and other discourse obligations, it has several limitations. The manager has to deal with issues from grounding to problem solving, including both task-level planning (booking flights) and discourse-level planning, as well as maintaining discourse context. There is no way for the interpretation and generation components to communicate directly. A more recent dialogue management architecture for TRIPS is shown in Figure 19.29. Here task-specific planning and problem solving is parceled out to a separate Task Manager. The Interpretation Manager recognizes rich dialog acts including problem-solving acts, and keeps discourse context. The Generation manager is sensitive to this knowledge. The Behavioral agent plans behavior based on its own goals and obligations as well as user utterances.

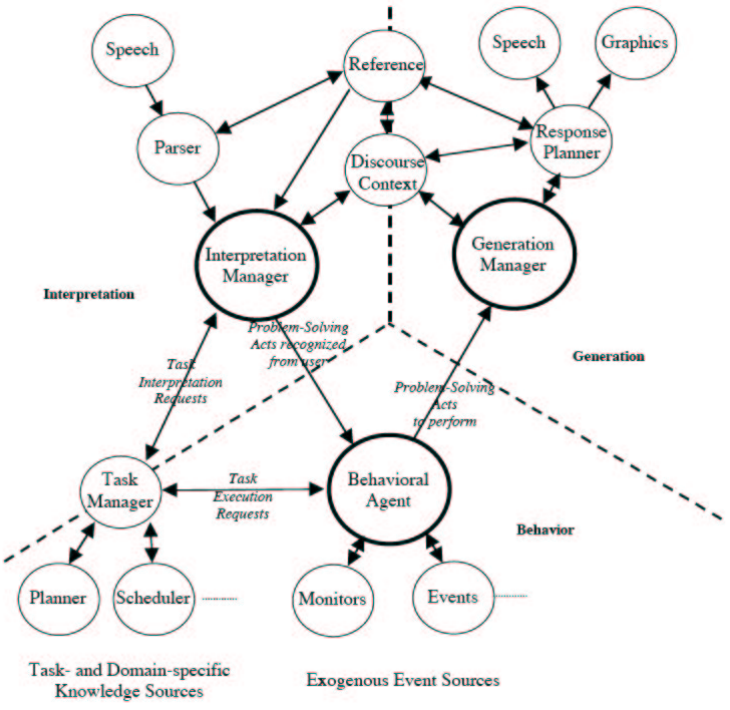

Figure 19.29 Dialogue System Architecture of the TRIPS system, after (Allen et al., 2001). PLACEHOLDER FIGURE.

19.10 SUMMARY

Conversational agents are a crucial speech and language processing application that are already widely used commercially. Research on these agents relies crucially on an understanding of human dialogue or conversational practices.

- Dialogue systems generally have 5 components: speech recognition, natural language understanding, dialogue management, natural language generation, and speech synthesis. They may also have a task manager specific to the task domain.
- Dialogue architectures for conversational agents include finite-state systems, **frame-based** production systems, Markov Decision Processes, and **BDI** (**belief-desire-intention**) models.
- Grounding and initiative are crucial human dialogue phenomena that must also be dealt with in conversational agents.
- Speaking in dialogue is a kind of action; these acts are referred to **dialogue acts**. Automatic interpretation of dialogue acts requires the use of lexical, syntactic, and prosodic knowledge.
- Human-human dialogue is another important area of dialogue, relevant especially for such computational tasks as **automatic meeting summarization**.
- Dialogue exhibits **intentional structure** in addition to the **informational structure**, including such relations as **dominance** and **satisfaction-precedence**.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

Early work on speech and language processing had very little emphasis on the study of dialogue. One of the earliest conversational systems, ELIZA, had only a trivial production system dialogue manager; if the human user's previous sentence matched the regular-expression precondition of a possible response, ELIZA simply generated that response (Weizenbaum, 1966). The dialogue manager for the simulation of the paranoid agent PARRY (Colby et al., 1971), was a little more complex. Like ELIZA, it was based on a production system, but where ELIZA's rules were based only on the words in the user's previous sentence, PARRY's rules also rely on global variables indicating its emotional state. Furthermore, PARRY's output sometimes makes use of script-like sequences of statements when the conversation turns to its delusions. For example, if PARRY's **anger** variable is high, he will choose from a set of "hostile" outputs. If the input mentions his delusion

topic, he will increase the value of his **fear** variable and then begin to express the sequence of statements related to his delusion.

The appearance of more sophisticated dialogue managers awaited the better understanding of human-human dialogue. Studies of the properties of human-human dialogue began to accumulate in the 1970's and 1980's. The Conversation Analysis community (Sacks et al., 1974; Jefferson, 1984; Schegloff, 1982) began to study the interactional properties of conversation. Grosz's (1977) dissertation significantly influenced the computational study of dialogue with its introduction of the study of substructures in dialogues (subdialogues), and in particular with the finding that "task-oriented dialogues have a structure that closely parallels the structure of the task being performed" (p. 27). The BDI model integrating earlier AI planning work (Fikes and Nilsson, 1971) with speech act theory (Austin, 1962; Gordon and Lakoff, 1971; Searle, 1975a) was first worked out by Cohen and Perrault (1979), showing how speech acts could be generated, and Perrault and Allen (1980) and Allen and Perrault (1980), applying the approach to speech-act interpretation.

See Walker and Whittaker (1990) and Chu-Carroll and Brown (1997) for more ways of defininining initiative in dialogue.

Models of dialogue as collaborative behavior were introduced in the late 1980's and 1990's, including the ideas of reference as a collaborative process (Clark and Wilkes-Gibbs, 1986), and models of **joint intentions** (Levesque et al., 1990), and **shared plans** (Grosz and Sidner, 1980). Related to this area is the study of **initiative** in dialogue, studying how the dialogue control shifts between participants (Walker and Whittaker, 1990; Smith and Gordon, 1997).

FIX: Add ATT work from 1990s into this history

FIX: Add more on survey chapters: McTear 2002, Sadek and DeMori

EXERCISES

19.1 List the dialogue act misinterpretations in the *Who's On First* routine at the beginning of the chapter.

19.2 Write a finite-state automaton for a dialogue manager for checking your bank balance and withdrawing money at an automated teller machine.

19.3 Dispreferred responses (for example turning down a request) are usually signaled by surface cues, such as significant silence. Try to notice the next time you or someone else utters a dispreferred response, and write down the utterance. What

are some other cues in the response that a system might use to detect a dispreferred response? Consider non-verbal cues like eye-gaze and body gestures.

19.4 When asked a question to which they aren't sure they know the answer, people use a number of cues in their response. Some of these cues overlap with other dispreferred responses. Try to notice some unsure answers to questions. What are some of the cues? If you have trouble doing this, you may instead read Smith and Clark (1993) which lists some such cues, and try instead to listen specifically for the use of these cues.

19.5 The sentence "*Do you have the ability to pass the salt?*" is only interpretable as a question, not as an indirect request. Why is this a problem for the BDI model?

19.6 Most universities require Wizard-of-Oz studies to be approved by a human subjects board, since they involve deceiving the subjects. It is a good idea (indeed it is often required) to "debrief" the subjects afterwards and tell them the actual details of the task. Discuss your opinions of the moral issues involved in the kind of deceptions of experimental subjects that take place in Wizard-of-Oz studies.

19.7 Implement a small air-travel help system. Your system should get constraints from the user about a particular flight that they want to take, expressed in natural language, and display possible flights on a screen. Make simplifying assumptions. You may build in a simple flight database or you may use a flight information system on the web as your backend.

19.8 Augment your previous system to work over the phone (or alternatively, describe the user interface changes you would have to make for it to work over the phone). What were the major differences?

19.9 Design a simple dialogue system for checking your email over the telephone. Assume that you had a synthesizer which would read out any text you gave it, and a speech recognizer which transcribed with perfect accuracy. If you have a speech recognizer or synthesizer, you may actually use them instead.

19.10 Test your email-reading system on some potential users. If you don't have an actual speech recognizer or synthesizer, simulate them by acting as the recognizer/synthesizer yourself. Choose some of the metrics described in the Methodology Box on page 22 and measure the performance of your system.

Allen, J. and Core, M. (1997). Draft of DAMSL: Dialog act markup in several layers. Unpublished manuscript.

Allen, J., Ferguson, G., Miller, B., and Ringger, E. (1995). Spoken dialogue and interactive planning. In *Proceedings ARPA Speech and Natural Language Workshop*, Austin, TX, pp. 202–207. Morgan Kaufmann.

Allen, J., Ferguson, G., and Stent, A. (2001). An architecture for more realistic conversational systems. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pp. 1–8. ACM Press.

Allen, J. and Perrault, C. R. (1980). Analyzing intention in utterances. *Artificial Intelligence*, *15*, 143–178.

Allwood, J. (1995). An activity-based approach to pragmatics. *Gothenburg Papers in Theoretical Linguistics*, *76*.

Allwood, J., Nivre, J., and Ahlsén, E. (1992). On the semantics and pragmatics of linguistic feedback. *Journal of Semantics*, *9*, 1–26.

Atkinson, M. and Drew, P. (1979). *Order in Court*. Macmillan, London.

Austin, J. L. (1962). *How to Do Things with Words*. Harvard University Press, Cambridge, MA.

Beckman, M. E. and Pierrehumbert, J. (1986). Intonational structure in English and Japanese. *Phonology Yearbook*, *3*, 255–310.

Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., and Winograd, T. (1977). GUS, A frame driven dialog system. *Artificial Intelligence*, *8*, 155–173.

Bois, J. W. D., Schuetze-Coburn, S., Cumming, S., and Paolino, D. (1983). Outline of discourse transcription. In Edwards, J. A. and Lampert, M. D. (Eds.), *Talking Data: Transcription and Coding in Discourse Research*, pp. 45–89. Lawrence Erlbaum, Hillsdale, NJ.

Bouwman, G., Sturm, J., and Boves, L. (1999). Incorporating confidence measure in the Dutch Train Timetable Information system developed in the ARISE project. In *IEEE ICASSP-99*, pp. 493–496. IEEE.

Bulyko, I., Kirchhoff, K., Ostendorf, M., and Goldberg, J. (2004). Error-sensitive response generation in a spoken langugage dialogue system. To appear in Speech Communication.

Bunt, H. (1994). Context and dialogue control. *Think*, *3*, 19–31.

Bunt, H. (2000). Dynamic interpretation and dialogue theory, volume 2. In Taylor, M. M., Neel, F., and Bouwhuis, D. G. (Eds.), *The structure of multimodal dialogue*, pp. 139–166. John Benjamins, Amsterdam.

Bunt, H. and Black, B. (2000). The ABC of computational pragmatics. In Bunt, H. C. and Black, W. (Eds.), *Computational Pragmatics: Abduction, Belief and Context*, pp. 1–46. John Benjamins, Amsterdam.

Carberry, S. (1990). *Plan Recognition in Natural Language Dialog*. MIT Press, Cambridge, MA.

Carletta, J., Dahlbäck, N., Reithinger, N., and Walker, M. A. (1997a). Standards for dialogue coding in natural language processing. Tech. rep. Report no. 167, Dagstuhl Seminars. Report from Dagstuhl seminar number 9706.

Carletta, J., Isard, A., Isard, S., Kowtko, J. C., Doherty-Sneddon, G., and Anderson, A. H. (1997b). The reliability of a dialogue structure coding scheme. *Computational Linguistics*, *23*(1), 13–32.

Chu-Carroll, J. (1998). A statistical model for discourse act recognition in dialogue interactions. In Chu-Carroll, J. and Green, N. (Eds.), *Applying Machine Learning to Discourse Processing. Papers from the 1998 AAAI Spring Symposium*. Tech. rep. SS-98-01, pp. 12–17. AAAI Press, Menlo Park, CA.

Chu-Carroll, J. and Brown, M. K. (1997). Tracking initiative in collaborative dialogue interactions. In *ACL/EACL-97*, pp. 262–270. Association for Computational Linguistics.

Chu-Carroll, J. and Carberry, S. (1998). Collaborative response generation in planning dialogues. *Computational Linguistics*, *24*(3), 355–400.

Chu-Carroll, J. and Carpenter, B. (1999). Vector-based natural language call routing. *Computational Linguistics*, *25*(3), 361–388.

Clark, H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Clark, H. H. and Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, *13*, 259–294.

Clark, H. H. and Wilkes-Gibbs, D. (1986). Referring as a collaborative process. *Cognition*, *22*, 1–39.

Cohen, M. H., Giangola, J. P., and Balogh, J. (2004). *Voice User Interface Design*. Addison-Wesley, Boston.

Cohen, P. R. and Perrault, C. R. (1979). Elements of a plan-based theory of speech acts. *Cognitive Science*, *3*(3), 177–212.

Cohen, P. and Oviatt, S. (1995). The role of voice input for human-machine communication. *Proceedings of the National Academy of Sciences*, *92*(22), 9921–9927.

Colby, K. M., Weber, S., and Hilf, F. D. (1971). Artificial paranoia. *Artificial Intelligence*, *2*(1), 1–25.

Cole, R. A., Novick, D. G., Vermeulen, P. J. E., Sutton, S., Fanty, M., Wessels, L. F. A., de Villiers, J. H., Schalkwyk, J., Hansen, B., and Burnett, D. (1997). Experiments with a spoken dialogue system for taking the US census. *Speech Communication*, *23*, 243–260.

Cole, R. A., Novick, D. G., Burnett, D., Hansen, B., Sutton, S., and Fanty, M. (1994). Towards automatic collection of the U.S. census. In *IEEE ICASSP-94*,

Adelaide, Australia, Vol. I, pp. 93–96. IEEE.

Cole, R. A., Novick, D. G., Fanty, M., Sutton, S., Hansen, B., and Burnett, D. (1993). Rapid prototyping of spoken language systems: The Year 2000 Census Project. In *Proceedings of the International Symposium on Spoken Dialogue*, Waseda University, Tokyo, Japan.

Core, M., Ishizaki, M., Moore, J. D., Nakatani, C., Reithinger, N., Traum, D., and Tutiya, S. (1999). The report of the third workshop of the Discourse Resource Initiative, Chiba University and Kazusa Academia Hall. Tech. rep. No.3 CC-TR-99-1, Chiba Corpus Project, Chiba, Japan.

Daly, N. A. and Zue, V. W. (1992). Statistical and linguistic analyses of F_0 in read and spontaneous speech. In *ICSLP-92*, Vol. 1, pp. 763–766.

Danieli, M. and Gerbino, E. (1995). Metrics for evaluating dialogue strategies in a spoken language system. In *Proceedings of the 1995 AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, Stanford, CA, pp. 34–39. AAAI Press, Menlo Park, CA.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Ford, C., Fox, B., and Thompson, S. A. (1996). Practices in the construction of turns. *Pragmatics*, *6*, 427–454.

Ford, C. and Thompson, S. A. (1996). Interactional units in conversation: syntactic, intonational, and pragmatic resources for the management of turns. In Ochs, E., Schegloff, E. A., and Thompson, S. A. (Eds.), *Interaction and Grammar*, pp. 134–184. Cambridge University Press, Cambridge.

Fraser, N. M. and Gilbert, G. N. (1991). Simulating speech systems. *Computer Speech and Language*, *5*, 81–99.

Good, M. D., Whiteside, J. A., Wixon, D. R., and Jones, S. J. (1984). Building a user-derived interface. *Communications of the ACM*, *27*(10), 1032–1043.

Goodwin, C. (1996). Transparent vision. In Ochs, E., Schegloff, E. A., and Thompson, S. A. (Eds.), *Interaction and Grammar*, pp. 370–404. Cambridge University Press, Cambridge.

Gordon, D. and Lakoff, G. (1971). Conversational postulates. In *CLS-71*, pp. 200–213. University of Chicago. Reprinted in Peter Cole and Jerry L. Morgan (Eds.), *Speech Acts: Syntax and Semantics Volume 3*, Academic, 1975.

Gorin, A., Riccardi, G., and Wright, J. (1997). How may i help you?. *Speech Communication*, *23*, 113–127.

Gould, J. D., Conti, J., and Hovanyecz, T. (1983). Composing letters with a simulated listening typewriter. *Communications of the ACM*, *26*(4), 295–308.

Gould, J. D. and Lewis, C. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM*, *28*(3), 300–311.

Grice, H. P. (1957). Meaning. *Philosophical Review*, *67*, 377–388. Reprinted in *Semantics*, edited by Danny D. Steinberg & Leon A. Jakobovits (1971), Cambridge University Press, pages 53–59.

Grice, H. P. (1975). Logic and conversation. In Cole, P. and Morgan, J. L. (Eds.), *Speech Acts: Syntax and Semantics Volume 3*, pp. 41–58. Academic Press, New York.

Grice, H. P. (1978). Further notes on logic and conversation. In Cole, P. (Ed.), *Pragmatics: Syntax and Semantics Volume 9*, pp. 113–127. Academic Press, New York.

Grosz, B. and Hirschberg, J. (1992). Some intonational characteristics of discourse structure. In *ICSLP-92*, Vol. 1, pp. 429–432.

Grosz, B. J. (1977). *The Representation and Use of Focus in Dialogue Understanding*. Ph.D. thesis, University of California, Berkeley.

Grosz, B. J. and Sidner, C. L. (1980). Plans for discourse. In Cohen, P. R., Morgan, J., and Pollack, M. E. (Eds.), *Intentions in Communication*, pp. 417–444. MIT Press, Cambridge, MA.

Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, *12*(3), 175–204.

Guindon, R. (1988). A multidisciplinary perspective on dialogue structure in user-advisor dialogues. In Guindon, R. (Ed.), *Cognitive Science And Its Applications For Human-Computer Interaction*, pp. 163–200. Lawrence Erlbaum, Hillsdale, NJ.

Heeman, P. A. and Allen, J. (1999). Speech repairs, intonational phrases and discourse markers: Modeling speakers' utterances in spoken dialog. *Computational Linguistics*, *25*(4).

Hemphill, C. T., Godfrey, J., and Doddington, G. R. (1990). The ATIS spoken language systems pilot corpus. In *Proceedings DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, pp. 96–101. Morgan Kaufmann.

Hinkelman, E. A. and Allen, J. (1989). Two constraints on speech act ambiguity. In *Proceedings of the 27th ACL*, Vancouver, Canada, pp. 212–219. ACL.

Hintikka, J. (1969). Semantics for propositional attitudes. In Davis, J. W., Hockney, D. J., and Wilson, W. K. (Eds.), *Philosophical Logic*, pp. 21–45. D. Reidel, Dordrecht, Holland.

Hirschberg, J. and Litman, D. J. (1993). Empirical studies on the disambiguation of cue phrases. *Computational Linguistics*, *19*(3), 501–530.

Hirschberg, J., Litman, D. J., and Swerts, M. (2001). Identifying user corrections automatically in spoken dialogue systems.. In *NAACL*.

Hirschberg, J. and Nakatani, C. (1996). A prosodic analysis of discourse segments in direction-giving monologues. In *Proceedings of ACL-96*, Santa Cruz, CA, pp.

286–293. ACL.

Hirschberg, J. and Pierrehumbert, J. (1986). The intonational structuring of discourse. In *ACL-86*, New York, pp. 136–144. ACL.

Hirschman, L. and Pao, C. (1993). The cost of errors in a spoken language system. In *EUROSPEECH-93*, pp. 1419–1422.

Issar, S. and Ward, W. (1993). Cmu's robust spoken language understanding system. In *Eurospeech 93*, pp. 2147–2150.

Jefferson, G. (1984). Notes on a systematic deployment of the acknowledgement tokens 'yeah' and 'mm hm'. *Papers in Linguistics*, pp. 197–216.

Jekat, S., Klein, A., Maier, E., Maleck, I., Mast, M., and Quantz, J. (1995). Dialogue Acts in VERBMOBIL verbmobil–report–65–95..

Jurafsky, D., Bates, R., Coccaro, N., Martin, R., Meteer, M., Ries, K., Shriberg, E., Stolcke, A., Taylor, P., and Van Ess-Dykema, C. (1997). Automatic detection of discourse structure for speech recognition and understanding. In *Proceedings of the 1997 IEEE Workshop on Speech Recognition and Understanding*, Santa Barbara, pp. 88–95.

Kamm, C. A. (1994). User interfaces for voice applications. In Roe, D. B. and Wilpon, J. G. (Eds.), *Voice Communication Between Humans and Machines*, pp. 422–442. National Academy Press, Washington, D.C.

Kita, K., Fukui, Y., Nagata, M., and Morimoto, T. (1996). Automatic acquisition of probabilistic dialogue models. In *ICSLP-96*, Philadelphia, PA, Vol. 1, pp. 196–199.

Kompe, R., Kießling, A., Kuhn, T., Mast, M., Niemann, H., Nöth, E., Ott, K., and Batliner, A. (1993). Prosody takes over: A prosodically guided dialog system. In *EUROSPEECH-93*, Berlin, Vol. 3, pp. 2003–2006.

Krahmer, E., Swerts, M., Theune, M., and Weegels, M. (1999). Error spotting in human-machine interactions. In *EUROSPEECH-99*, Budapest, pp. 1423–1426.

Labov, W. and Fanshel, D. (1977). *Therapeutic Discourse*. Academic Press, New York.

Ladd, D. R. (1996). *Intonational Phonology*. Cambridge Studies in Linguistics. Cambridge University Press.

Landauer, T. K. (Ed.). (1995). *The Trouble With Computers: Usefulness, Usability, and Productivity*. MIT Press, Cambridge, MA.

Levesque, H. J., Cohen, P. R., and Nunes, J. H. T. (1990). On acting together. In *AAAI-90*, Boston, MA, pp. 94–99. Morgan Kaufmann.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, *8*, 11–23.

Levinson, S. C. (1983). *Pragmatics*. Cambridge University Press, Cambridge.

Levow, G.-A. (1998). Characterizing and recognizing spoken corrections in human-computer dialogue.. In *COLING-ACL*, pp. 736–742.

Litman, D. J. (1985). *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*. Ph.D. thesis, University of Rochester, Rochester, NY.

Litman, D. J. and Allen, J. F. (1987). A plan recognition model for subdialogues in conversation. *Cognitive Science*, *11*, 163–200.

Litman, D. J., Hircshberg, J. B., and Swerts, M. (2000). Predicting automatic speech recognition performance using prosodic cues. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL)*, Seattle, Washington, pp. 218–225.

Litman, D. J. and Pan, S. (2002). Designing and evaluating an adaptive spoken dialogue system. *User Modeling and User-Adapted Interaction*, *12*(2-3), 111–137.

Litman, D. J. and Silliman, S. (2004). Itspoke: An intelligent tutoring spoken dialogue system. In *Proceedings of HLT/NAACL-2004*.

Litman, D. J., Walker, M. A., and Kearns, M. S. (1999). Automatic detection of poor speech recognition at the dialogue level. In *ACL-99*, pp. 309–316. Association for Computational Linguistics.

Lochbaum, K. E. (1998). A collaborative planning model of intentional structure. *Computational Linguistics*, *24*(4), 525–572.

Lochbaum, K. E., Grosz, B. J., and Sidner, C. L. (1990). Models of plans to support communication: An initial report. In *AAAI-90*, Boston, MA, pp. 485–490. Morgan Kaufmann.

Mast, M., Kompe, R., Harbeck, S., Kießling, A., Niemann, H., Nöth, E., Schukat-Talamazzini, E. G., and Warnke, V. (1996). Dialog act classification with the help of prosody. In *ICSLP-96*, Philadelphia, PA, Vol. 3, pp. 1732–1735.

Meteer, M. and Iyer, R. (1996). Modeling conversational speech for speech recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania, pp. 33–47. ACL.

Miller, S., Bobrow, R., Ingria, R., and Schwartz, R. (1994). Hidden understanding models of natural language. In *Proceedings of the 32nd ACL*, Las Cruces, NM, pp. 25–32. ACL.

Miller, S., Fox, H., Ramshaw, L., and Weischedel, R. (2000). A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL)*, Seattle, Washington, pp. 226–233.

Miller, S., Stallard, D., Bobrow, R., and Schwartz, R. (1996). A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting*

of the ACL, Santa Cruz, California, pp. 55–61.

Moore, J. D. and Pollack, M. E. (1992). A problem for RST: The need for multi-level discourse analysis. *Computational Linguistics*, *18*(4), 537–544.

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.

Nielsen, J. (1992). The usability engineering life cycle. *IEEE Computer*, 12–22.

Norman, D. A. (1988). *The Design of Everyday Things*. Doubleday, New York.

Oviatt, S., Cohen, P. R., Wang, M. Q., and Gaston, J. (1993). A simulation-based research strategy for designing complex NL sysems. In *Proceedings DARPA Speech and Natural Language Workshop*, Princeton, NJ, pp. 370–375. Morgan Kaufmann.

Oviatt, S., MacEachern, M., and Levow, G.-A. (1998). Predicting hyperarticulate speech during human-computer error resolution. *Speech Communication*, *24*, 87–110.

Passonneau, R. and Litman, D. J. (1993). Intention-based segmentation: Human reliability and correlation with linguistic cues. In *Proceedings of the 31st ACL*, Columbus, Ohio, pp. 148–155. ACL.

Perrault, C. R. and Allen, J. (1980). A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, *6*(3-4), 167–182.

Pieraccini, R., Levin, E., and Lee, C.-H. (1991). Stochastic representation of conceptual structure in the ATIS task. In *Proceedings DARPA Speech and Natural Language Workshop*, Pacific Grove, CA, pp. 121–124. Morgan Kaufmann.

Pierrehumbert, J. and Hirschberg, J. (1990). The meaning of intonational contours in the interpretation of discourse. In Cohen, P. R., Morgan, J., and Pollack, M. (Eds.), *Intentions in Communication*, pp. 271–311. MIT Press, Cambridge, MA.

Pierrehumbert, J. (1980). *The Phonology and Phonetics of English Intonation*. Ph.D. thesis, MIT.

Polifroni, J., Hirschman, L., Seneff, S., and Zue, V. (1992). Experiments in evaluating interactive spoken language systems. In *Proceedings DARPA Speech and Natural Language Workshop*, Harriman, New York, pp. 28–33. Morgan Kaufmann.

Power, R. (1979). The organization of purposeful dialogs. *Linguistics*, *17*, 105–152.

Reichman, R. (1985). *Getting Computers to Talk Like You and Me*. MIT Press, Cambridge, MA.

Reithinger, N., Engel, R., Kipp, M., and Klesen, M. (1996). Predicting dialogue acts for a speech-to-speech translation system. In *ICSLP-96*, Philadelphia, PA, Vol. 2, pp. 654–657.

Reithinger, N. and Klesen, M. (1997). Dialogue act classification using language models. In *EUROSPEECH-97*, Vol. 4, pp. 2235–2238.

Russell, S. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ. Second edition.

Sacks, H., Schegloff, E. A., and Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language*, *50*(4), 696–735.

Sag, I. A. and Liberman, M. (1975). The intonational disambiguation of indirect speech acts. In *CLS-75*, pp. 487–498. University of Chicago.

Samuel, K., Carberry, S., and Vijay-Shanker, K. (1998). Dialogue act tagging with transformation-based learning. In *COLING/ACL-98*, Montreal, Vol. 2, pp. 1150–1156. ACL.

San-Segundo, R., Montero, J., Ferreiros, J., Còrdoba, R., and Pardo, J. (2001). Designing confirmation mechanisms and error recovery techniques in a railway information system for Spanish. In *In Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue, Aalborg, Denmark*.

Schegloff, E. A. (1968). Sequencing in conversational openings. *American Anthropologist*, *70*, 1075–1095.

Schegloff, E. A. (1982). Discourse as an interactional achievement: Some uses of 'uh huh' and other things that come between sentences. In Tannen, D. (Ed.), *Analyzing Discourse: Text and Talk*, pp. 71–93. Georgetown University Press, Washington, D.C.

Schegloff, E. A. (1988). Presequences and indirection: Applying speech act theory to ordinary conversation. *Journal of Pragmatics*, *12*, 55–62.

Schegloff, E. A., Jefferson, G., and Sacks, H. (1977). The preference for self-correction in the organization of repair in conversation. *Language*, *53*, 361–382.

Searle, J. R. (1975a). Indirect speech acts. In Cole, P. and Morgan, J. L. (Eds.), *Speech Acts: Syntax and Semantics Volume 3*, pp. 59–82. Academic Press, New York.

Searle, J. R. (1975b). A taxonomy of illocutionary acts. In Gunderson, K. (Ed.), *Language, Mind and Knowledge, Minnesota Studies in the Philosophy of Science*, Vol. VII, pp. 344–369. University of Minnesota Press, Amsterdam. Also appears in John R. Searle, *Expression and Meaning: Studies in the Theory of Speech Acts*, Cambridge University Press, 1979.

Seneff, S. (1995). TINA: A natural language system for spoken language application. *Computational Linguistics*, *18*(1), 62–86.

Shriberg, E., Bates, R., Taylor, P., Stolcke, A., Jurafsky, D., Ries, K., Coccaro, N., Martin, R., Meteer, M., and Ess-Dykema, C. V. (1998). Can prosody aid the automatic classification of dialog acts in conversational speech?. *Language and Speech (Special Issue on Prosody and Conversation)*, *41*(3-4), 439–487.

Shriberg, E., Wade, E., and Price, P. (1992). Human-machine problem solving using spoken language systems (SLS): Factors affecting performance and user satisfaction. In *Proceedings DARPA Speech and Natural Language Workshop*, Harriman, New York, pp. 49–54. Morgan Kaufmann.

Singh, S. P., Litman, D. J., Kearns, M. J., and Walker, M. A. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the njfun system.. *J. Artif. Intell. Res. (JAIR)*, *16*, 105–133.

Smith, R. W. 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, V. L. and Clark, H. H. (1993). On the course of answering questions. *Journal of Memory and Language*, *32*, 25–38.

Stalnaker, R. C. (1978). Assertion. In Cole, P. (Ed.), *Pragmatics: Syntax and Semantics Volume 9*, pp. 315–332. Academic Press, New York.

Stifelman, L. J., Arons, B., Schmandt, C., and Hulteen, E. A. (1993). VoiceNotes: A speech interface for a hand-held voice notetaker. In *Human Factors in Computing Systems: INTERCHI '93 Conference Proceedings*, Amsterdam, pp. 179–186. ACM.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martina, R., Meteer, M., and Ess-Dykema, C. V. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, *26*, 339–371.

Stolcke, A., Shriberg, E., Bates, R., Coccaro, N., Jurafsky, D., Martin, R., Meteer, M., Ries, K., Taylor, P., and Van Ess-Dykema, C. (1998). Dialog act modeling for conversational speech. In Chu-Carroll, J. and Green, N. (Eds.), *Applying Machine Learning to Discourse Processing. Papers from the 1998 AAAI Spring Symposium*. Tech. rep. SS-98-01, Stanford, CA, pp. 98–105. AAAI Press.

Stolcke, A. and Shriberg, E. (1996). Automatic linguistic segmentation of conversational speech. In *ICSLP-96*, Philadelphia, PA, pp. 1005–1008.

Suhm, B. and Waibel, A. (1994). Toward better language models for spontaneous speech. In *ICSLP-94*, Vol. 2, pp. 831–834.

Swerts, M., Litman, D., and Hirschberg, J. (2000). Corrections in spoken dialogue systems. In *ICSLP-00*, Beijing, China.

Taylor, P., King, S., Isard, S., and Wright, H. (1998). Intonation and dialog context as constraints for speech recognition. *Language and Speech*, *41*(3-4), 489–508.

Traum, D. R. and Allen, J. (1994). Discourse obligations in dialogue processing. In *Proceedings of the 32nd ACL*, Las Cruces, NM, pp. 1–8. ACL.

Waibel, A. (1988). *Prosody and Speech Recognition*. Morgan Kaufmann, San Mateo, CA.

Walker, M., Kamm, C., and Litman, D. (2001). Towards developing general models of usability with PARADISE. *Natural Language Engineering: Special Issue on Best Practice in Spoken Dialogue Systems*, *6*(3).

Walker, M., Passonneau, R., Rudnicky, A., Aberdeen, J., Boland, J., Bratt, E., Garofolo, J., Hirschman, L., , Le, A., Lee, S., Narayanan, S., Papineni, K., Pellom, B., Polifroni, J., Potamianos, A., Prabhu, P., Rudnicky, A., Sanders, G., Seneff, S., Stallard, D., and Whittaker, S. (2002). Cross-site evaluation in DARPA Communicator: The June 2000 data collection. submitted.

Walker, M. A., Fromer, J. C., and Narayanan, S. (1998). Learning optimal dialogue strategies: a case study of a spoken dialogue agent for email. In *Proceedings of the 17th international conference on Computational linguistics*, pp. 1345–1351. ACL.

Walker, M. A., Langkilde, I., Wright, J., Gorin, A., and Litman, D. (2000). Learning to predict problematic situations in a spoken dialogue system: Experiments with how may i help you?. In *Proceedings of ANLP-NAACL Conference*. Seattle.

Walker, M. A., Litman, D. J., Kamm, C. A., and Abella, A. (1997). PARADISE: A framework for evaluating spoken dialogue agents. In *ACL/EACL-97*, Madrid, Spain, pp. 271–280. ACL.

Walker, M. A., Maier, E., Allen, J., Carletta, J., Condon, S., Flammia, G., Hirschberg, J., Isard, S., Ishizaki, M., Levin, L., Luperfoy, S., Traum, D., and Whittaker, S. (1996). Penn multiparty standard coding scheme: Draft annotation manual. `www.cis.upenn.edu/~ircs/discourse-tagging/newcoding.html`.

Walker, M. A. and Rambow, O. C. (2002). Spoken language generation. *Computer Speech and Language*, *16*(3-4), 273–281.

Walker, M. A. and Whittaker, S. (1990). Mixed initiative in dialogue: An investigation into discourse segmentation. In *Proceedings of the 28th ACL*, Pittsburgh, PA, pp. 70–78. ACL.

Ward, W. and Issar, S. (1994). Recent improvements in the cmu spoken language understanding system. In *ARPA Human Language Technologies Workshop*, Plainsboro, N.J.

Warnke, V., Kompe, R., Niemann, H., and Nöth, E. (1997). Integrated dialog act segmentation and classification using prosodic features and language models. In *EUROSPEECH-97*, Vol. 1, pp. 207–210.

Weizenbaum, J. (1966). ELIZA – A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Woszczyna, M. and Waibel, A. (1994). Inferring linguistic structure in spoken language. In *ICSLP-94*, Yokohama, Japan, pp. 847–850.

Xu, W. and Rudnicky, A. I. (2000). Task-based dialog management using an agenda. In *ANLP/NAACL Workshop on Conversational Systems*, Somerset, New Jersey, pp. 42–47. Association for Computational Linguistics.

Yankelovich, N., Levow, G.-A., and Marx, M. (1995). Designing SpeechActs: Issues in speech user interfaces. In *Human Factors in Computing Systems: CHI '95 Conference Proceedings*, Denver, CO, pp. 369–376. ACM.

Yngve, V. H. (1970). On getting a word in edgewise. In *CLS-70*, pp. 567–577. University of Chicago.

Young, S. (2002). The statistical approach to the design of spoken dialogue systems. Tech. rep. CUED/F-INFENG/TR.433, Cambridge University Engineering Department, Cambridge, England.

Zue, V., Glass, J., Goodine, D., Leung, H., Phillips, M., Polifroni, J., and Seneff, S. (1989). Preliminary evaluation of the VOYAGER spoken language system. In *Proceedings DARPA Speech and Natural Language Workshop*, Cape Cod, MA, pp. 160–167. Morgan Kaufmann.