CMU-CS-79-144

# An Anatomy of Graceful Interaction in Spoken and Written Man-Machine Communication

Phil Hayes
Raj Reddy

13 September 1979

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-78-C-1551.

## Abstract

There have recently been a number of attempts to provide natural and flexible interfaces to computer systems through the medium of natural language. While such interfaces typically perform well in response to straightforward requests and questions within their domain of discourse, they often fail to interact gracefully with their users in less predictable circumstances. Most current systems cannot, for instance: respond reasonably to input not conforming to a rigid grammar; ask for and understand clarification if their user's input is unclear; offer clarification of their own output if the user asks for it; or interact to resolve any ambiguities that may arise when the user attempts to describe things to the system.

We believe that graceful interaction in these and the many other contingencies that can arise in human conversation is essential if interfaces are ever to appear cooperative and helpful, and hence be suitable for the casual or naive user, and more habitable for the experienced user. In this paper; we attempt to circumscribe graceful interaction as a field for study, and identify the problems involved in achieving it.

To this end we decompose graceful interaction into a number of relatively independent skills: skills involved in parsing elliptical, fragmented, and otherwise ungrammatical input; in ensuring robust communication; in explaining abilities and limitations, actions and the motives behind them; in keeping track of the focus of attention of a dialogue; in identifying things from descriptions, even if ambiguous or unsatisfiable; and in describing things in terms appropriate for the context. We claim these skills are necessary for any type of graceful interaction and sufficient for graceful interaction in a certain large class of application domains. None of these components is individually much beyond the current state of the art, and we outline the architecture of a system that integrates them all. Thus, we propose graceful interaction as an idea of great practical utility whose time has come and which is ripe for implementation. We are currently implementing a gracefully interacting system along the lines presented; the system will initially deal with typed input, but is eventually intended to accept natural speech.

## Table of Contents

## Introduction

A great deal of interest has recently been shown in providing natural and flexible computer interfaces through systems capable of engaging in a dialogue with a human being in more or less natural language. This interest is embodied in numerous systems including GUS [2], LIFER [20], PAL [37], PARRY [30], and PLANES [40], and in the work of Codd [6], Grosz [14], and others. Such systems typically respond accurately and appropriately to straightforward requests and questions or otherwise uneventful dialogue within their domain of discourse. Compared to human beings, however, the performance of current natural language interfaces appears quite rigid and fragile. Most current systems cannot, for instance: respond reasonably to input not conforming to a rigid grammar; ask for and understand clarification if their user's input is unclear; offer clarification of their own output if the user asks for it; or interact to resolve any ambiguities that may arise when the user attempts to describe things to the system. A dialogue system cannot hope to interact naturally and gracefully with its users, unless it can cope with these and the many other contingencies, so common in ordinary human conversation.

Graceful interaction, then, involves dealing appropriately with anything a user happens to say, rather than just those inputs in the mainstream of a conversation. Even though little attention has been paid to it in work to date (PARRY [30] being the main exception), we believe that graceful interaction is essential in making natural language interfaces (for both typed and spoken input) appear cooperative and helpful to their users, and thus in making computer systems more accessible to casual or naive users, and more pleasant and generally habitable for the more experienced user.

In this paper, we claim that the ability to interact gracefully depends on a number of relatively independent skills: skills involved in parsing elliptical, fragmented, and otherwise ungrammatical input; in ensuring robust communication; in explaining abilities and limitations, actions and the motives behind them; in keeping track of the focus of attention of a dialogue; in identifying things from descriptions, even if ambiguous or unsatisfiable; and in describing things in terms appropriate for the context. While none of these components of graceful interaction has been entirely neglected in the literature, no single current system comes close to having most of the abilities and behaviours we describe, and many are not possessed by any current systems. This will become clear in our discussion of each component. We believe, however, that none of these components is individually much beyond the current state of the art, and later in the paper we will present the architecture of a system that combines them all.

Thus, we believe that graceful interaction is an idea of great practical utility whose time

has come, and which is ripe for implementation. This paper is an attempt to circumscribe graceful interaction as a field of study, to identify the problems that need to be solved, and to present the design of a system that, for a suitably restricted domain, is capable of truly graceful interaction. We are currently implementing a gracefully interacting system along the lines presented; the system will initially deal with typed input, but is eventually intended to accept natural speech.

## 1. Components of Graceful Interaction

Graceful interaction is not a single monolithic skill. Rather, it seems to be composed of a number of diverse abilities and behaviours. In the succeeding sections, we will describe a set of abilities and behaviours that appear to be essential for graceful interaction. Although this set contains necessary components of graceful interaction, it is probably not sufficient for graceful interaction in general; however, we believe it provides a good working basis from which to build gracefully interacting systems, and in particular, those which provide a simple service in a restricted domain (e.g. telephone directory assistance, restaurant reservations, computer mail services); we will define this class a little more carefully in Section 4.2.

The components of graceful interaction we describe are based on phenomena observable in naturally occuring human dialogues. We believe it is very important for a gracefully interacting system to conduct a dialogue in as human-like a way as possible; if the strategies the system employs for clarifying its incomprehensions of the user, resolving ambiguous descriptions supplied by the user, etc. are not the same as those a human would use in the same situation, then the user will feel that the interaction is not natural, and hence not graceful. Furthermore, most of the components of graceful interaction we have gleaned from human conversations involve *cooperation* between speaker and listener; if the user tries to employ any of these techniques in his interaction with the system, an inability of the system to cooperate will appear to him most ungraceful. Thus the freedom we wish to give the user to express himself exactly as he wishes requires that a gracefully interacting system be able to deal with dialogue problems in much the same way as a human does.

Unfortunately, the aim of being as human-like as possible must be tempered by the limited potential for comprehension of any forseeable computer system. Until a solution is found to the problems of organizing and using the range of world knowledge possessed by a human, practical systems will only be able to comprehend a small amount of input, typically within a specific domain of expertise. Graceful interaction must, therefore, supplement its simulation of human conversational ability with strategies to deal naturally and gracefully with input that is not fully understood, and, if possible, to steer a conversation back to the system's home ground.

We propose seven components of graceful interaction. This set is probably not sufficient, nor are the boundaries between the several components completely watertight; arguments could be made for drawing them in different places. However, it would be hard to argue that any of the seven were unimportant for graceful interaction. The components we propose are:

- **Robust communication:** The set of strategies needed to ensure that a listener receives a speakers utterance, and interprets it correctly.

- **Flexible parsing:** The ability to deal with naturally used natural language, with all the ellipses, idioms, grammatical errors, and fragmentary utterances it can contain.

- **Domain knowledge:** Not strictly-speaking a component of graceful interaction, but a prerequisite for all the other components.

- **Explanation facility:** The ability of the system to explain what it can and cannot do, what it has done, what it is trying to do, and why, both for response to direct questions, and as a fall-back when communication breaks down.

- **Focus mechanisms:** The ability to keep track of what the conversation is about, as the items under discussion shift; this is important for the resolution of ellipsis and anaphora, as well as for continuity in the conversation.

- **Identification from descriptions:** The ability to recognize an object from a description; this involves the ability to pursue a clarifying dialogue if the original description is not clear.

- **Generation of descriptions:** The ability to generate descriptions that are appropriate for the context, and that satisfy requirements imposed by the other components, especially robust communication.

While none of these components of graceful interaction have been entirely neglected in the literature, no single current system comes close to having most of the abilities and behaviours we describe, and many are not possessed by any current systems.

In what follows, we will consider each component in detail; in particular, in each case we will discuss:

- the natural language and dialogue phenomena on which the component is based;

- the abilities and behaviours needed by a computer dialogue system to participate gracefully in a conversation containing these phenomena (including those differing from humans, but necessitated by those limitations of current dialogue systems that are unlikely to be removed in the near future);

- the extent to which any current systems contain the component (including any reasons why their basic structure or methods might preclude their incorporation

of the component, and the implications of this for the design of systems which try to implement the component).

In these discussions, we will in general make no distinction between spoken and typed language; this reflects our view that most of the principles of graceful interaction are the same whether the medium of communication is speech or text. In cases where it does make a difference, mainly those involving problems with speech that do not arise for typed input, we will note the difference explicitly. The discussions will be illustrated by example conversation fragments; these conversations are intended to be typical of those that might occur between a gracefully interacting system and its user; accordingly, we have set them in simple service domains of the type mentioned above (see Section 4.2 for a more precise definition of simple service domains). In fact, we have used two domains of discourse in our examples: an internal directory assistance system, and a restaurant reservation system. The examples chosen are based in part on protocols of actual conversations (with two human participants), and in part on conversations invented to illustrate particular points.

## 2. Robust Communication

During the course of a conversation, it is not uncommon for people to misunderstand or fail to understand each other. Such failures in communication do not usually cause the conversation to break down; rather, the participants are able to resolve the difficulty, usually by a short clarifying sub-dialogue, and continue with the conversation from where they left off. Current computer systems are unable to take part in such clarifying dialogues, or resolve communication difficulties in any other way. As a result, when such difficulties occur, a computer dialogue system is unable to keep up its end of the conversation, and a complete breakdown is likely to result; this fragility lies in stark and unfavourable contrast to the robustness of human dialogue.

In this section, we discuss the cooperative techniques that humans use to overcome difficulties in communication, and the ways in which these techniques can be made available to computer dialogue systems.

### 2.1. Introduction - The Principle of Implicit Confirmation

Every time a human being speaks to another in an attempt to communicate something, he faces three obstacles to getting his message across:

- the listener may not receive the message;
- the listener may receive the message but be unable to interpret all or part of it;

- the listener may interpret the message incorrectly.

Although these difficulties do not occur in the majority of attempts at communication, they are not uncommon in ordinary human conversation and arise completely unpredictably. If one of them does occur, the listener will not receive the message that the speaker intended to convey with potentially serious consequences for the conversation. Despite these unpredictable errors in communication, human dialogue is extremely robust; people almost always manage to get their message across. This robustness does not stem from the elimination or minimization of such errors, but rather from a set of techniques, based on cooperation between speaker and listener, that humans use to detect, recover from, and correct the errors that occur. It is these techniques and their application to graceful interaction that we will be considering in this and the following subsections. The present subsection, in particular, discusses a convention, tacitly agreed by participants in a human dialogue, on which all the techniques are based.

Since a speaker typically has no direct way of telling whether his listener has received his message correctly, detection of communication difficulties must rest on some convention of acknowledgement, commonly agreed upon by speaker and listener. One possible convention is for the listener to explicitly acknowledge everything that the speaker says. This is essentially the technique employed for communication between networks of computers in which analogous communication problems can arise. While this convention ensures extremely accurate communication, constant explicit acknowledgement would be too tedious for humans. Instead, humans use a convention which requires much less overhead at the cost of occaisonally allowing inaccurate communication. We call this convention the principle of <u>implicit confirmation</u>, and can state it as follows:

> The speaker assumes his message has been received by the listener, and received correctly, unless the listener indicates otherwise.

This assumption of communication on the speaker's part is analogous to the assumption on the speaker's part that the listener will follow any shift in focus he may make, as noted by Grosz [15].

While this approach is obviously very efficient when there is no difficulty with communication, it places a large burden on the listener. Unless the listener can determine that he has not received the message correctly, the error will go undetected; the conversation will continue uncorrected and may become quite confused. Marx brothers' comedies exploit such confusions to good effect. This is the basic reason that human techniques for robust communication can sometimes fail. Fortunately, the listener normally has enough expectations about the sorts of things the speaker might say to make this a rare

occurrence.

In order for its interaction to appear natural, a gracefully interacting system must use this same principle of implicit confirmation in its conversation with its user. Given the necessarily limited linguistic abilities of current systems, a gracefully interacting system is, in fact, liable to run into more communication errors in the form of miscomprehension and incomprehension than is a human, and if the system tries to use another convention (such as explicit acknowledgement) to deal with these errors, it will seem most ungraceful to the user. Furthermore, the user will naturally employ implicit confirmation, and unless the system understands this convention and cooperates in it, inaccurate communication, and considerable frustration on the part of the user is liable to result. As far as we know, no current dialogue system conducts its dialogues in accordance with this principle of implicit confirmation.

Although we have been using the terms, "speaker" and "listener", everything we have said applies equally to dialogue typed at a terminal. Typed dialogues are just as susceptible to incomprehension and miscomprehension as spoken dialogues; the only difference is in the sources of such problems in communication. In typed dialogues, words are not usually misrecognized as they can be with speech; on the other hand, one cannot make spelling errors in spoken dialogue. In what follows, we will continue to blur the distinction between the two forms of dialogue; our discussion will be ostensibly of spoken dialogue, but all the points we make will apply equally to typed dialogue unless otherwise noted.

In the next subsection, we will discuss details of how the speaker can tell if his message fails to be received at all; the two subsequent subsections discuss two simple methods by which the listener can indicate his lack of comprehension or lack of confidence in his comprehension. In all three cases we will comment on the extent to which the techniques discussed are used in current dialogue systems, and any modifications that are likely to be required in a practical gracefully interacting system, particularly those required by the limitations of the current state of the art in language understanding. The final subsection is a summary.

### 2.2. Implicit Acknowledgements

In this section we consider how the speaker can tell if his message was received at all, whether correctly or incorrectly, and what he can do if it was not received.

As the principle of implicit confirmation states, the speaker assumes his message was received unless the listener indicates otherwise. How can the listener indicate that he has not received a message which we are assuming he does not know has been sent? Obviously, he cannot do this actively or explicitly; instead, he does it tacitly by not doing anything i.e. by

not replying. To put it another way, the speaker expects a reply, and if he does not receive one will assume that his message has not been received at all. This expectation provides a way consistent with the principle of implicit confirmation of detecting communication errors in which the listener fails to receive any of the speaker's message.

In general human dialogue, replies need not be linguistic; for instance, the listener might simply perform an action that the speaker requested. However, for the purposes of graceful interaction, in which conversations are either typed or spoken without any visual communication (as over a telephone), we may confine ourselves to purely linguistic replies. The reply can come either when the speaker has finished delivering his message and has paused to await a reply (typed input systems operate exclusively this way) or by an interruption before the end of the message. If no reply is forthcoming, the speaker will assume that his message has not been received.

One option with a message that has not been received is to repeat it; the more usual and, if repetition fails, eventually necessary strategy is to assume that there is some fault in the channel of communication, and to try either to re-establish communication or to confirm that the channel is defunct. When the speaker is in the presence of the listener, this attempt might be a shouted "can you hear me"; the listener might be asleep or deaf. Over the telephone, a much more common situation, the attempt uses phrases such as "Hello!", "Are you there?", "I can hear you, can you hear me?". If the original speaker receives suitable replies to the channel checking utterances, the substantive part of the dialogue can proceed. If the channel is indeed defunct, the dialogue is at an end. Similar but more conventional forms are used at the beginning of a dialogue to ensure that the channel is indeed open and available for communication.

We have said that a reply to a message is enough to convince the speaker of the message that the message was indeed received. Are there any constraints on the form of the reply, or will any reply do? Clearly any reply of the channel-checking form will not confirm communication, but will actively indicate failure of the message to get across:

S: The number for Joe Smith is 5267
U: Hello! Are you there?
S: Yes! Can you hear me?

The appropriate response is to participate in a channel checking dialogue with the user.

Apart from this case, there are very few other responses which would not be taken to imply reception (correct or incorrect) of the message. It is not necessary to obtain an answer to a question; counter questions and even quite distantly related statements serve as replies:

U: What is the number for Joe Smith?
S: Do you know the address?
U: Sorry! I mean Bill Smith.

If the listener gives a reply that appears completely inappropriate to the original speaker, he is more likely to assume that the listener received the original message incorrectly, rather than not at all.

A gracefully interacting system should, of course, conform to the human conventions about implicit acknowledgements, i.e. it should reply to all of its user's utterances within a short period (possibly giving a time filling reply or request to "hold on" if it cannot give a substantive reply soon enough), and it should expect its user to reply to it in a similarly short time. If this expectation is not met, it should be able to initiate and complete a channel-checking dialogue, and if possible, return to its original utterance. Of course, it must also be able to suspend this kind of "time-out" expectation if the user asks it to wait, and naturally there is no need to limit the "patience" of such a system very tightly.

The importance of replying to the user within a short time of his input has been stressed in the LIFER system [20]. The immediate replies, however, took the form of progress reports on the processing of the input, and essentially served as time-fillers. Descriptions of other systems such as PARRY [30] and PLANES [40] have stressed the need for rapid replies, but these systems do not attempt to monitor the speed of their own replies and produce a time-filling reply if it is too slow. One version of the SCHOLAR system [4] did, however,apologize if it took too long to respond. No systems to our knowledge are able to conduct a channel-checking dialogue.

### 2.3. Explicit Indications of Incomprehension

The principle of implicit confirmation requires a listener to give an explicit indication whenever he fails to comprehend a message or suspects that he may have received a message incorrectly. In this section we deal with the most straightforward way a listener can do this: by asking the speaker what he said or meant. Such questions vary according to how much of the original utterance the listener thinks he understood, how informative he tries to be about the nature of the problem, and how much he wants to influence the speaker's subsequent reply.

The least informative way of indicating incomprehension, and indirectly of asking the speaker to repeat his message is through a phrase such as "I beg your pardon?" or "What was that?" If the lack of understanding was caused by a transient problem such as noise on the channel or inattention on the part of the speaker, this strategy may result in a second and successful attempt at communication. If, on the other hand, the failure was caused by a

non-transient problem such as the listener's unfamiliarity with the words or construction used (uncommon with human beings, but likely to occur frequently with computer systems), a second attempt at communication is no more likely to be successful than the first. Since the original speaker is given no clue to the nature of the difficulty the listener is having in understanding what he is saying, he is unable to correct that difficulty except by changing what he is saying in a hit and miss fashion. Successful communication is more likely to result if the listener can provide some clues to the nature of the problem, and the conventions of human dialogue make it easy to do so. Indeed, we believe that the provision of information to the original speaker about where and what he is failing to communicate is fundamental to the extraordinary robustness of human conversation. Without the ability to zero in on problems, human conversation could not achieve robustness in the apparently effortless way that it does.

The specificity of the clues the listener can provide to the speaker about the nature and location of a communication difficulty depends on the degree to which the listener failed to understand the utterance. If he understood nothing at all, the only clue he can provide is what he expected the speaker to say:

S: This is directory assistance.
U: <GARBLE>
S: I didn't quite catch that; can I get a number for you?

If he has understood something, he can show what he has understood while indicating that he has not understood fully:

U: What is the number for <GARBLE>?
S: Whose number did you want?

U: What is the number for Jim <GARBLE>?
S: Jim who?

This allows the original speaker to concentrate on transmitting only the part of the message that was not understood, making "Smith", for instance, a suitable reply in the second example. If the listener has understood enough to narrow the possibilities down to a small number he can list them.

U: What is the number for <?>ill Smith?
S: Did you say Bill Smith or Phil Smith?

This strategy allows the original speaker to solve the communication problem with a phrase like "the second one", which is presumably much less susceptible to error than repetition of the part of the communication that originally caused the error ("Bill" or "Phil"). In general, the information given by these various strategies to the original speaker allows him to concentrate on imparting the information that did not get across, and thus makes for much

greater directedness and robustness in the resolution of communication difficulties. As a final strategy, if the listener has produced an interpretation for the utterance, but is uncertain of that interpretation because of noise or because the interpretation is incongruous, he can ask explicitly if his complete interpretation is correct.

The way in which incomprehension is indicated can influence the form of the subsequent clarification. This fact is of particular importance for a gracefully interacting system with limited power of comprehension. It is not sufficient for such a system merely to indicate incomprehension, or even to indicate it informatively, so that its user is made clearly aware of the nature of its difficulty in comprehension; it should also indicate its incomprehension in a form most likely to elicit a clarification from the user that the system can comprehend. For instance, as Codd [6] points out, questions of the form "What do you mean by ..." are liable to elicit a description on a level much too sophisticated for such a system. Thus if the system does not understand "extension" in:

What is the extension for Jim Smith?

it is unwise to reply:

What is an extension?

but better to ask:

Do you want Jim Smith's number or his address?[1]

Both replies indicate the problem word exactly, and are thus in some sense equally informative about the nature of the problem in comprehension, but the second is preferable since it encourages the user to use either "number" or "address" in his reply, and hence give the system a good chance of understanding it.

While a gracefully interacting system should try to shape its users responses, the form of the responses cannot be guaranteed. In particular, the system cannot pose multiple choice questions to the user and expect the user always to pick one of the choices given, as do many current systems including Codd's RENDEZVOUS system [6]. Multiple choice questions are contrary to the whole spirit of graceful interaction and sequences of them would soon frustrate a user. A gracefully interacting system can and should be able to present alternatives to the user as in the last example, but it must be prepared for the user to express his choice in his own way or to come up with an entirely different option.

Do you want Jim Smith's telephone number or his address?

---

[1] It is, of course, highly desirable that the system should note the reply to this question, and remember it if the user ever says the word, extension, again, since repetition of such a question would surely be very frustrating to a user. Single word learning of this sort has been discussed by Carbonell in connexion with his Politics system [3]. In general, the problems of learning from mistakes and adapting to the idiosyncracies of individual users are aspects of graceful interaction we have chosen not to deal with.

Examples of reasonable responses include:

The number.
The first one.
Both.
I want to phone him.
Do you know where he is now?
I want his social-security number.
I want the number, but I meant Joe Smith.

This raises the possibility that the user's clarification will also be misunderstood, and of a sequence of misunderstandings and clarifications failing to converge. In such cases, a gracefully interacting system will have to become more and more explicit about the nature of its problems and its limitations and rely on the user to accommodate himself (see Section 5.6.

In short, a gracefully interacting system must be able to express its own incomprehension of or uncertainty about what the user said, and be able to deal with the user's reports of his own problems in comprehending the system. While most systems can indicate incomprehension in some way, most are uninformative and undirective in the senses we have discussed. As far as we know, no current system can respond appropriately to explicit complaints of incomprehension by its user. Clearly, formulating adequate replies to such complaints would require a system to keep track of the things it said and the reasons for saying them.

### 2.4. Echoing and the Use of Fragmentary Recognition

One restriction associated with explicit indications of incomprehension is the expectation of a reply. In this section we discuss a technique, called echoing, which the listener can use to confirm his interpretation of a message, but which does not require a reply from the original speaker if the interpretation is correct; a reply is needed if the interpretation turns out to be incorrect. Echoing is thus an efficient way of confirming interpretations of which the listener is fairly sure, but is less useful for cases in which the listener is less certain. We also mention another way of reducing the number of explicit indications of incomprehension - by using fragmentary recognitions of utterances as the basis for complete interpretations. Such interpretations can never, of course, be guaranteed to be correct, and they should be confirmed through echoing.

If the original speaker receives a message indicating that his message has not been completely or confidently understood, he will normally try to clarify or confirm his original message. In particular, an explicit request for confirmation cannot be answered implicitly:

U: What is the number <?>ter Smith?
S: Did you say Walter Smith?
U: Can you also give me his address?

This example is unnatural unless the first speaker says "yes" before asking for the address. It would become tedious to answer too many explicit requests for confirmation, so it is fortunate that human dialogue provides a method of asking for confirmation in a way that allows an affirmative response to be given implicitly. When the listener wants to be sure that his interpretation of the speaker's utterance (or more commonly part of it) is correct, he has only to echo his interpretation. If the original speaker does not comment on the echo, then he implicitly confirms it. Thus:

U: What is the number for <?>ter Smith?
S: Walter Smith ...
S: His number is 5592.
U: 5592. Thank you.
S: You're welcome.

Of course, the original speaker still has the option of confirming the echo explicitly; and if the echo is incorrect, he must indicate that explicitly.

Direct echoes of this type are useful because they can be confirmed implicitly, thus allowing a listener to verify his understanding without disrupting the flow of the conversation by a clarifying sub-dialogue. Nevertheless, a direct echo is still somewhat disruptive by its very presence; if it were not for its verifying function, it would be completely superfluous to the conversation. There is, however, another verifying technique which avoids even this small amount of disruption. The technique, which we will call indirect echoing, is to incorporate the assumption to be verified into the listener's next utterance in the normal flow of the conversation. For example:

U: What is the <GARBLE> for Jim Smith?
S: The number for Jim Smith is 2597.

In this example, S is able to determine from the linguistic and non-linguistic context that <GARBLE> is almost certainly "number". S then incorporates this assumption into his reply as an indirect echo. The net effect is much the same as for a direct echo ("the number"); if U, the original speaker, does not comment on this indirect echo it is implicitly confirmed, and if he wishes to correct it explicitly, he may do so. The essential difference from a direct echo is the absence of any utterance by S outside the natural flow of the conversation. Interestingly enough, this seems to deny the original speaker the opportunity to confirm the assumption explicitly.

The form of an echo need not always correspond to the form of the utterance which is echoed, but can be a paraphrase of it. The following examples show paraphrased echoes of the direct and indirect types respectively.

U: I want to contact Roger Smith.
S: The number for Roger Smith ...
S: It's 2597.

U: I want to <GARBLE> Jim Smith.
S: The number for Jim Smith is 2957.

A number of question answering systems, including RENDEZVOUS [6] and COOP [23, 28], have adopted the policy of presenting the user with a paraphrased version of each input, generated from the system's internal representation of that input. This gives the user a chance to find and correct any misinterpretations made by the system during its interpretation of his input. However, the unconditional generation of paraphrases cannot be considered graceful interaction, and quickly become tedious for the user. Direct echoes, whether paraphrases or not, should only be generated when there is some significant degree of uncertainty in the interpretation. Selective paraphrase generation along these lines, which of course involves deciding when uncertainty exists, has been attempted by Carbonell in his MICS system [3].

In addition to its use for checking the correctness of interpretations, echoing, especially direct echoing, can also serve as a time filler. In an analysis of protocols of directory assistance conversations, we found that it was common to echo the name being asked for while the number was being found (which typically took several seconds). This is perhaps explainable by a listener's need (discussed in Section 2.2) to reply in order to convince the speaker that his utterance has been received at all. It appears that echoes in this case served just the same purpose as a phrase such as: "Just a second!", indicating that the communication had been received, but that there would be a delay before the real reply was given. It is interesting to note that a full echo of the input is rarely given in such time-filling echoes; In the protocols, people echoed only the name as opposed to any other portions of the request for a number. In doing this they are perhaps conforming to the conventions of the clarifying type of echo, by echoing the part of the input least predictable in the given context, and therefore most liable to misinterpretation.

In order to conduct natural dialogues, and to avoid frustrating its user with a stream of trivial complaints of incomprehension, a gracefully interacting system should conform to the human conventions of echoing; i.e. it should be able to issue echoes (direct or indirect) when it is uncertain in its comprehension, or when appropriate as time-fillers, and should be able to monitor and make use of any corrections its user offers in response. It must also be constantly on the watch for echoes by the user of what it says, and should be ready to issue corrections as necessary. As far as we know, none of these components of echoing has ever been implemented in a dialogue system.

A way more radical than echoing of avoiding explicit requests for clarification is to ignore uncomprehended but inessential elements of what the user said. In general, it is impossible to tell whether uncomprehended input is essential or not, simply because it is uncomprehended. However, there are a number of heuristic strategies that the system can try. First, if the incomprehensible element is small enough and is contextually determined as a function word or "noise phrase", it can be ignored. It is often unimportant whether a determiner is definite or indefinite or which preposition is used. Certain larger segments can be ignored through a key-word approach. For instance, as was found in work on GUS [2], users are likely to offer explanations for their desires, so that if any unrecognizable sequence can be found which begins with "because", "since", etc., it is not unreasonable to ignore that sequence as an irrelevant explanation.

A more general way of deciding whether or not a partial comprehension is sufficient is to decide whether the recognized components can be combined into something that the system would find it reasonable for the user to say. For example, if a travel agent system could extract a city and a date from what a user said to it, it could build a request for a reservation to that city on that date. The combination of recognized fragments is, in fact, the basis of the question answering ability of Waltz's PLANES system [40], and other work on the combination of such fragments has been done by Fox and Mostow [11].

None of the above methods of ignoring certain unrecognized elements of an input are close to being foolproof; they can, and sometimes will, produce fundamental errors in comprehension. For this reason it is important that a gracefully interacting system make clear what assumptions it is making and what the results of its comprehension really are. It would be extremely tedious for the user if the system always made explicit requests for confirmation of everything it was unsure of, particularly if the language capabilities of the system were less than complete as we might expect for systems in the forseeable future. A heavy use of direct echoes is little better, since the user would scarcely feel that excessively parrot-like behaviour was very graceful. A more palatable alternative is for the system to incorporate the assumptions it makes into its reply through an indirect echo. For example, if all a travel agent system could extract from:

I am interested in paying a visit to Pittsburgh on or around May 17

was "Pittsburgh" and "May 17", then its reply could be:

About what time on May 17 would you like to go to Pittsburgh?

This indirect echo, if uncorrected by the user, will confirm both the system's interpretation of the fragments, as well as its assumptions about their relationship (it could have been *from* Pittsburgh instead of *to* Pittsburgh). The notion of always making the system's assumptions

explicit is present in the work of Codd [6]; however, Codd suggested that his system should present its understanding of each of the user's requests for explicit approval by the user before proceeding to satisfy the request.

### 2.5. Summary of Robust Communication

We can summarize the techniques of robust communication, and their implications for gracefully interacting systems as follows:

- Human dialogue ensures accurate communication without the overhead of explicit acknowledgement by using the principle of implicit confirmation, which allows the speaker to assume his message has been received correctly by the listener, unless the listener indicates otherwise.

- A speaker expects his listener to reply. If the listener does not reply within a very short time, the original speaker will assume his message has not got across, and initiate a dialogue to check the channel of communication. From the point of view of the principle of implicit confirmation, the absence of a reply serves as a tacit indication by the listener that he has not received the speaker's communication. A gracefully interacting system must be able to: conform to the convention of immediate reply, if necessary with a time-filling reply; initiate a channel-checking dialogue if its user does not reply; and participate in a channel-checking dialogue started by the user.

- The simplest way for the listener to indicate that he has not received the speaker's message correctly is to ask him what he said or meant, while possibly at the same time indicating the nature and source of the problem in communication. Such an indication requires a reply by the original speaker. Different ways of asking the question tend to constrain the reply to a greater or lesser extent. The necessarily limited linguistic abilities of a dialogue system suggest that it should ask the most constraining questions it can in such situations.

- When a listener thinks he has understood what the speaker said, but is not quite sure, he can echo (the tentative part) of what he thought he heard, either directly in a separate utterance, or indirectly by incorporating the echo into his next utterance in the normal flow of the conversation. If the original speaker does not correct the echo, it is implicitly confirmed. Echoing thus allows the listener to confirm his interpretation, without requiring a reply from the speaker. A dialogue system can avoid many trivial questions by using this technique, and being ready to accept any corrections that are given. In any case, a system must monitor its user's echoes of what it says, and be prepared to correct any mistakes.

- Other explicit indications of incomprehension, with their requirements for replies, can be avoided if the system guesses what the user said on the basis of partial or fragmentary recognition of the input. Assumptions made in this way must be echoed to avoid confusion.

# 3. Flexible Parsing

The language used in naturally occurring dialogues between humans deviates in many ways from commonly accepted standards of grammaticality. There may be incorrect prepositions, inflexions, or lack of agreement; utterances may be elliptical, fragmentary, or idiomatic; there may be omitted or repeated words or phrases; utterances may be broken off and restarted at any point. These deviations occur most frequently in spoken dialogues, but are also sufficiently common in typed conversations that any computer system which attempts to engage humans in graceful natural language dialogue of the typed or spoken variety must be able to deal with them.

Unfortunately, most current parsing methodologies are not well suited to the analysis of such deviant input. Most systems analyse linguistic input in accordance with their expectations, grammatical and otherwise, which cannot practically be extended to cover all, or even most, of the possible deviations; input failing to conform to these expectations, by even so much as a single word is typically totally incomprehensible to such systems. In short, most current parsing systems are quite inflexible in the face of deviant input.

By **flexible parsing** we mean parsing which can deal with input deviating from grammatical norms and/or the grammatical expectations of the system doing the parsing. Given the common occurrence of grammatical deviations in naturally occuring dialogues, flexible parsing is clearly of prime importance for graceful interaction. In the following subsection we describe various ways in which language in natural dialogues can and does deviate from the grammatical norm; in a second subsection, we consider current parsing techniques, the way in which they have been used to deal (or fail to deal) with these deviations, and their potential in this regard.

## 3.1. Grammatical Deviations

In this section we describe some of the most common types of deviations from standard grammar that arise in natural human conversations: idioms, fragmentary utterances, omissions, repetitions, insertion of noise phrases, small grammatical errors, and ellipsis.

**Idioms:** Everyday language contains many idioms, i.e. phrases whose interpretation cannot be obtained by using the components of the phrase in the usual way ("a wild goose chase" has very little to do with geese). Even though the structure of idioms is often grammatical, it is unhelpful to parse them, since by definition their meaning cannot be obtained from their components; it is necessary to interpret them as a whole.

Many other phrases while not strictly speaking idiomatic can often conveniently be dealt with as a whole. It is possible, for instance, to analyse:

Could you give me the number for Joe Smith

as a conditional question about ability, and from that to recognize it as an indirect speech act making a request for information. On the other hand, a directory assistance system would probably find it more convenient to interpret "could you give me" directly as a request for information.

**Fragmentary utterances:** Humans are adept at piecing together fragments of an utterance to come up with an interpretation of the utterance as a whole. For people, the need for such fragmentary recognition occurs mainly when noise of some sort or lack of clarity in pronunciation has made it impossible for them to hear all of an utterance, or when a speaker has actually spoken in disjointed fragments. In the case of computer systems, a more frequent need for fragmentary recognition might arise through ignorance of vocabulary or constructions. In either case, dealing with such fragmentary input, as humans seem able to do, requires an ability to extract the largest recognizable fragments from the input, and to use those fragments to construct a reasonable interpretation.

In many cases, if it is possible to construct a reasonable interpretation of recognized fragments at all, then that interpretation will be the correct one. For example, if the fragments "give me" and "the number for Joe Smith" were recognized out of:

Would you be so kind as to give me your listing of the number for Joe Smith

then the obvious interpretation would be the right one. On the other hand, fragments can be pieced together incorrectly, as in the same limited understanding of:

I asked you to give me the number for Joe Smith, but I meant Fred.

Thus, allowing fragmentary recognition raises the question of when the uncomprehended portion of an utterance can be safely ignored, and when it cannot. There is surely no solution to this problem that is guaranteed always to be correct, but as discussed in Section 2.4, the techniques of robust communication are helpful when an interpretation built from fragments turns out to be wrong.

**Omissions, repetitions, and noise phrases:** It is possible to omit and repeat words or phrases or insert noise phrases into an utterance without significantly decreasing its intelligibility as in:

What the the number for lemme see Joe Smith?

Of course, function words have the least effect, but in the proper context content words or

phrases may be omitted or repeated without loss of comprehension. Omission and repetition is much more of a problem in speech input than in typed input (though it does occur in the latter). The noisiness of a speech signal can often effectively cause omission of words or parts of words, (particularly small function words); while repetition commonly occurs as a person repeats (or worse, replaces) words, or sequences of words as he is producing an utterance as in:

What is er could er you g...give me the number er the extension for Joe Smith?

This written representation does not indicate the prosodic features - rising and falling intonations, pauses, and stress - present in such utterances; these features may be very important in human comprehension of utterances containing such repetition, breaking-off, and restarting, as well as in comprehension of more fluent utterances.

**Grammatical Errors:** It is not uncommon through carelessness or ignorance, to make mistakes in the use of language without materially affecting comprehensibility. Examples are incorrect tenses, lack of agreement between subject and verb, using the wrong preposition, and, for typed input, misspelling. Humans appear to be remarkably unaffected by these errors in their language understanding, often not even noticing that errors have been made.

**Ellipsis:** It is not uncommon in dialogue to say things that would be meaningless outside the context of the dialogue:

U: What is the number for Mr. Smith?
S: Do you mean Joe Smith or Fred Smith?
U: Joe

In this interchange, "Joe" is said to be an ellipsis of the more complete "I mean Joe Smith". Needless to say, humans are seldom confused by this such omissions, and are able to use the linguistic and non-linguistic context to understand the elliptical utterance without difficulty. There is no clear indication whether human resolution of ellipsis involves the construction of a more complete linguistic form.

### 3.2. Implications for Language Analysers

Having presented some of the ways in which naturally used language can deviate from grammatical norms, we turn to the question of how current parsing techniques can cope with such deviations, both in practice and in theory. We will consider a whole range of parsers, including traditional top-down left-to-right parsers using fixed grammars of both the syntactic and semantic variety, conceptual parsers, pattern-matching parsers of various degrees of sophistication, and parsers developed for use specifically with spoken input. There is no intention, however, to survey the field of parsing, so references to existing

systems will be cursory, and will assume prior knowledge on the part of the reader.

The most common parsing technique in use today is *top-down left-to-right* parsing based on a fixed grammar. This technique is usually implemented by an augmented transition network (ATN) of the type developed by Woods [45], or some close variation; such parsers can parse in terms of either general grammatical categories as in the LUNAR system [46], or in terms of categories having some significance within a restricted domain of discourse as in the LADDER system [33].

Parsers of this type are extremely fragile; they typically fail to produce any sort of parse if their input deviates from their grammar by so much as a single word. This property obviously makes them extremely unsuitable for dealing with input with missing or repeated words, grammatical errors, or with fragmentary input.

These parsers are so fragile because of the depth-first way in which their top-down left-to-right algorithms explore the set of potential parses. When a partial parse fails because the next word does not fall into one of the categories allowed by the grammar at that point, the failure is taken to mean that the parser made an incorrect choice earlier; that parse path is abandoned; and a different choice is made at an earlier choice point. Clearly, unless the input conforms exactly to one of the possibilities allowed by the grammar, no parse at all will result. This fail and back-up procedure is so fundamental to the way in which such parsers search their often very large space of possibilities that it is very difficult to modify their algorithms to deal with repeated words, incorrect endings, etc.. Weischedel and Black [41] have made some progress in this direction, by arranging to relax predicates enforcing such grammatical constraints as noun-verb agreement whenever a straightforward parse fails, but these tactics can deal with only a relatively limited class of grammatical deviations.

An approach which avoids much of this fragility and opens up the possibility of fragmentary recognition involves the use of a number of *specialist subgrammars*, each capable of recognizing a description of one particular type of entity or analysing one particular type of construction. While each subgrammar is applied in the same top-down, left-to-right fashion as before, they are applied independently, so that failure of one to find a parse does not affect the performance of the others. This is the approach taken by Waltz in his PLANES [40] system, which has a subgrammar for each type of entity known to the system; if an input contains references to several different entities known to the system, then each of them is analysed quite independently of the others.

While splitting up the recognition in this way ensures that a single repeated or omitted word or grammatical mistake will not wreck the entire parse, the parsing of each

subcomponent still suffers from the same fragility as before, and there are the extra problems of deciding at which point in the input to apply each subgrammar, and of one subgrammar analysing input which should have been analysed by a different one. In addition, while it is desirable to be able to parse fragments if the need arises, it is also desirable to parse utterances as a whole whenever possible; if an utterance is always parsed as a set of fragments, it is always necessary to construct a complete interpretation out of the interpretations of the fragments. In the case of Waltz's system, the domain was sufficiently constrained that the interpretation of a set of fragments was always unique, but in general this will, of course, not be true.

More possibility for flexibility in this area of missing and extra words and grammatical errors seems to be offered by the *conceptual parser* of Riesbeck [32]. Since a parse by that system is organized around and directed by the meaning of the key action or state in the sentence being parsed, the parse could presumably be made less sensitive to the presence of the correct function words. In practice, this potential flexibility does not seem to have been greatly exploited, and the system still relies heavily on finding, for example, the correct prepositions in prepositional phrases.

A quite different style of analysis is based on *pattern-matching* or rewrite rules. In the simplest type of pattern-matching approach, interpretation of input is obtained by matching it as a whole against a set of patterns of words. This was essentially the way in which input was analysed by many early AI language processing programs, such as ELIZA [42] and SIR [31]. Early versions of PARRY [8] used an approach barely more sophisticated than this, and pattern matching (of structures more complex than words) was the basis of Wilks machine translation system [43].

Flexibility in pattern-matching parsing can be obtained by flexibility in the matching process. Many of the simpler pattern matching systems used variables in their patterns which could match arbitrary strings, but while this resulted in systems which could analyse a very wide range of input, the level of analysis was correspondingly shallow. More sophisticated forms of flexibility could presumably be obtained by allowing partial matches of patterns, with perhaps restrictions to ensure that some content words were included in the match; it is not hard to see how this could deal with repeated and omitted words, and in some cases grammatical errors. Surprisingly enough, this potential has not been widely exploited; The more advanced version of PARRY [30], for instance, seems to rely on exact matching of very general patterns, rather than inexact matching of specific patterns to deal with deviant input. Nevertheless, pattern-matching parsing seems to offer the greatest potential for dealing with grammatical mistakes, and missing and repeated words.

A further advantage of pattern matching for the purposes of flexible parsing is its ability to handle *idioms* and fixed phrases. In fact, since the structure of idiomatic phrases is by definition unhelpful in their interpretation, some sort of wholistic approach such as pattern-matching seems almost obligatory for their interpretation. Indeed, a number of more traditional parsers including the one for the LUNAR system [46] have a distinct pre-processing stage in which idioms are recognized by a pattern-matching process.

Pattern matching does, however, have its limitations. If one tries to analyse complete utterances by single patterns, there will be commonalities between the patterns corresponding to the regularities of expression in the domain of discourse. It is these regularities that the more usual type of grammar is designed to account for. The solution within a pattern-matching system is to introduce rewrite rules, substituting the results in place of what is matched. In this way, patterns which account for regularities in the use of auxiliary verbs can be combined with patterns which recognize specific idioms to produce a language analyser, based on pattern-matching, but capable of coping with regularities in a non-redundant way. The power of this approach has been shown in more recent work on PARRY [30].

This uncertainty about where fragments begin and end makes any top-down approach with either a left-to-right or right-to-left directionality inappropriate for fragmentary recognition. A bottom-up approach is indicated. Again, a hierarchical pattern-matching approach would seem well suited on these grounds, but it has not been used in practice this way. Because of the special uncertainties of their medium, several *speech parsers*, including those of the HEARSAY-II [18] and HWIM [47] systems, have dealt with the fragmentary recognition problem by applying strict grammars in a bottom-up non-directional fashion. The extra robustness achieved by this technique is, however, bought at the price of considerably increased search effort. These systems also suggest that top-down recognition still has a role to play since the expansion of already recognized fragments in accordance with the grammar creates hypotheses about what exists on either side of the fragments, and if these hypotheses can be validated in a top-down manner, efficiency is greatly enhanced.

Once fragments have been recognized, it is necessary to *construct an interpretation from the fragments.* In the PLANES system, it is assumed that there is always exactly one way of combining the interpretation of the fragments to produce a meaningful interpretation; this strategy works because of a highly limited domain of discourse. Other, more general, work on fragment combination by Fox and Mostow [11] deals also with situations in which conflicting fragments have been found.

Most parsers do not deal with *ellipsis*, the omission from an utterance of details that

context can provide (see also Section 6.3). The problem is tackled in a limited way for parsers built by LIFER [20]. Any input not recognized in the normal way is assumed to be an ellipsis of a sentence structurally analogous to the last previously recognized input, and an attempt is made to parse the input according to each of the possible structural analogies. The parser of the GUS system [2] deals with ellipsis in reply to a question (e.g. Q: "When do you want to go?" A: "ten", instead of A: "I want to go at ten"), by inserting an otherwise unrecognizable input into a template generated from the question asked, ("I want to go at ..." for "When do you want to go?"), and then parsing the filled-out template in the normal way. Neither mechanism can cope with more general forms of ellipsis.

This general approach to ellipsis is based on the assumption that the "complete" form of the elliptical input must be discovered before the input can be analysed. An alternative approach is to treat elliptical inputs as complete within their context. In practical terms this means predicting possible ellipses according to the context, and attempting to parse them directly from the input. Such a direct approach avoids the possibly unnecessary complication of enclosing the (presumably) important part of an input in a "completing" context only to have to unwrap it again in the analysis. Carbonell [3] has attempted to deal with ellipsis in essentially this way in the context of question answering systems. Of course, not all ellipses can be predicted easily enough for it to be efficient to recognize them this way; an alternative approach, untried as far as we know, might be to deal with such ellipses in the same way as fragmentary input.

### 3.3. Summary of Flexible Parsing

The language used in naturally occuring dialogues often deviates from strict grammatical norms; common types of deviation include: omitted and repeated words, incorrect tenses, inflexions and prepositions, idiomatic, elliptical, and fragmentary utterances. Current parsing systems typically do not deal with these types of deviant input. In the case of parsers which apply a strict grammar to their input in a top-down left-to-right fashion, whether that grammar is semantically based or purely syntactic, there seem to be fundamental difficulties involved in dealing with input which does not conform to the grammar. The majority of attacks on deviant input seem to have taken place in the context of pattern-matching parsers, and although this type of parser is less suited to the regularities of language than a more traditional top-down parser, there is reason to hope that these difficulties can be overcome. Pattern-matching parsers are very well suited to deal with idiomatic input, and also lend themselves to the kind of bottom-up analysis that appears necessary to deal with fragmentary input.

The points we have made in this section about the flexible parsing of naturally occuring

language can be itemized as follows:

- Strict top-down left-to-right parsing schemes, such as those based on ATNs, appear irredeemably unsuited to deal with small grammatical deviations such as repeated or omitted words, incorrect inflexions, etc..

- These problems can be localized, but not overcome through the use of small specialist subgrammars.

- The basic approach of conceptual parsers - fitting an input into a predefined conceptual framework suggested by the main action or state of an input - seems potentially much better able to deal with grammatical irregularities, but has not yet been exploited in this way.

- Pattern-matching parsers are in a similar situation; they have the potential to deal with grammatical irregularities through flexible partial matching schemes, but this potential has been little exploited.

- Idioms and other fixed phrases are best handled by a pattern-matching approach.

- Fragmentary recognition requires a basically bottom-up approach.

- Speech-parsers demonstrate the advantages in efficiency afforded by allowing top-down strategies to be used after a context has been established by bottom-up methods.

- Only limited forms of ellipsis are handled by current parsing systems; all current methods rely on filling in the ellipsed components, and parsing the completed input; a little explored alternative strategy is to recognize ellipses as complete inputs, either by prediction or in the same way as other fragments.

## 4. Domain Knowledge

In this section we discuss the importance to a gracefully interacting system of knowledge about its domain of discourse; consider a class of domains and related tasks that appear especially appropriate for gracefully interacting systems, and that will figure heavily in our discussions of the remaining components of graceful interaction; and finally, describe a method of knowledge representation appropriate for gracefully interacting systems in such domains.

### 4.1. The Role of Domain Knowledge

A computer system cannot interact gracefully with its user unless it has substantial knowledge about the domain that the interaction concerns. Such domain knowledge is not, however, a clearly separable component of graceful interaction like those we have discussed; rather, it is a prerequisite or underpinning for the other components. A parser, for instance,

requires knowledge of how words and the phrasings in which they are used relate to the underlying domain; and for the purposes of robust communication, it is important to know what are the important, information-bearing parts of a user's utterance, and which parts can be safely ignored, even if not fully understood; an explanation facility clearly requires knowledge of the abilities and mode of operation of the system itself.

For the two components of graceful interaction already discussed: robust communication and flexible parsing, we were able to describe the component in a more or less domain independent way. Even though examples were necessarily restricted in domain, it is not hard to see how the principles and requirements we established apply to virtually any domain. Nor did our discussion depend at all on the way in which the requisite domain knowledge was represented; we were able simply to assume that it was available.

Neither of these simplifying assumptions holds for the four components discussed in the following four sections: explanation facility, goals and focus mechanisms, identification from descriptions, and language generation. While the components are relevant to and important for an equally wide-range of domain as the preceding components, they will be significantly different according to the particular type of domain involved. In discussing them, we will try to give a perspective on them for a range of domains; but we will concentrate our attention on one (quite broad) class of domains that we will call simple-service domains, and which include both the directory assistance and restaurant reservations domains that we have been following.

In the following two subsections, we will describe this class of domains a little more carefully, and consider appropriate representations for knowledge about such domains.

### 4.2. Simple Services

Many of the simple services provided in current society, especially those offered over the telephone, require in essence only that the customer or client identify certain entities to the person providing the service; these entities are parameters of the service, and once they are identified the service can be provided. We will call tasks which can be cast in this framework simple services. Examples of such services are directory assistance (the person, business or organization whose number is desired is the entity that must be identified) and restaurant reservations (party size, time, date must be identified); in fact, airline and any other type of reservation falls into this mould, along with requests for weather information, current stock market prices, and any other "current value" requests for information. An additional simple service, which for obvious reasons is not available from humans, is provided by an electronic mail system; the parameters for sending an item of electronic mail, for instance, are the

source and destination, plus the (unanalysed) body of the message.

Examples of non-simple service domains for gracefully interacting systems include assembly tasks as studied by a group at Stanford Research Institute [39] in which an expert supervises a novice assembling a mechanical device (the obvious role for a gracefully interacting system here is the expert), or more generally any supervisory or instructional task. A complete secretarial service would also be beyond the realm of simple services.

We have singled out simple service domains for two reasons: their commonness, and their tractability. It is clear from the above examples that simple service systems are very common in the real world, so a solution to the graceful interaction problem which was restricted to simple service domains would still be extremely useful. Furthermore, we believe that the graceful interaction problem is tractable in a relatively straightforward way for simple service systems. In particular, we believe that the set of components of graceful interaction that we are proposing is sufficient for systems operating in simple service domains. We still claim that our proposed components are necessary for all gracefully interacting systems, whether operating in simple service domains or not, but other non-simple service domains may require extra skills.

### 4.3. Representations for Simple Service Domains

Now we turn to the question of how to represent knowledge about simple service domains. The use of conversational systems in simply service domains is far from novel. Two important examples are the GUS system [2], which made round-trip plane reservations between pairs of cities in California, and the PAL system [37], which scheduled appointments from specifications of participants, meeting place, time, etc..[1] Both systems used frames to organize and represent their domain knowledge.

Frames are a method of knowledge representation, named and popularized by Minsky [29], which seem particularly well suited to simple service domains because their structure reflects the natural structure of a simple service: a service specified through the description of a limited set of entities which serve as parameters to the service.

A frame is a representation of some entity in terms of the entities which make it up; a physical object can be represented in terms of its parts, an event can be represented in terms of its participants, and a more complex event can be represented in terms of its

[1] Actually, PAL is not an interactive system. It accepts a monologue which specifies all the information necessary to schedule a meeting. Nevertheless, its domain is simple service and many of the same problems, particularly in the area of focus, arise as for a gracefully interacting system.

sub-events. The components of a frame are called its slots. The use of frames in simple service systems is thus straightforward; the entities which the user must describe to the system correspond to slots in a frame which represents the service as a whole. The system performs the service by filling the slots according to the user's descriptions, plus possibly performing some manipulations on the completed frame.

The use of frames provides a number of clear-cut advantages for graceful interaction in simple service domains. First, a frame is a declarative data structure, so that is easy for a system to manipulate its components in whatever order and however often it desires; this allows a simple service system to deal with a user's descriptions of slots in whatever order they are presented, and to go back and change the entity filling a slot if the user changes his mind. Furthermore, the declarative nature of a frame allows the system easily to keep track of what has been accomplished so far in a conversation, and what still has to be done. If the user fails to volunteer a description for any required slot, the resulting hole in the frame is a signal for the system to take the initiative and ask the user for a description of an appropriate entity. In the same way, a full complement of filled slots can be the system's cue to perform the service and attempt to terminate the conversation. As we will see in Section 6.3, a slot (or the goal of filling it) also provides a good representation for what the attention of the system and user is directed to at any given time. In short, knowledge about a simple service in the form of a frame provides a useful way of organizing the acquisition of information necessary to perform the service.

The slots in a frame are themselves declarative data structures, and so can contain pointers to information other than their actual fillers. In particular, slots commonly refer to other frames which describe the entities that may fill them; thus, the person slot of a directory assistance frame might refer to a frame describing a person with slots for surname, first name, title, etc.; in turn a surname might be described by a frame with slots for individual letters. These references to other frames serve as type information to help a system decide in which slot a given description should be placed. In addition, the potential they provide to represent the grain of descriptions to essentially arbitrary levels of fineness (Jim Smith; Jim; a person whose first name begins with 'J') is important for the focus and identification from description components, as we shall see.

Another type of reference found in slots is to procedures to be invoked when a filler is found for a slot or when an attempt is made to find a filler. This technique, called procedural attachment, was heavily exploited by the GUS system to allow transmission of fillers from one slot to another and to provide special strategies to fill selected slots. As an example, consider an electronic mail system which, like ARPAnet mail, provides a separate FROM and SENDER field; if no FROM is specified, then it is the same as the person composing the

message; if a different FROM is specified, then SENDER is the same as the composer. A procedure to transmit the default contents of FROM to SENDER if a different FROM were specified would clearly be very useful in this case.

A third type of information that could be attached to slots in temporary information used in establishing the contents of the slot. For instance, if a system had only partially understood a user's description of a slot, it could store in the slot what it had understood, plus perhaps some hypotheses about what it had not understood, the better to understand the user's reply to its request for clarification. In addition, it is sometimes necessary to maintain temporary information about alternative slot fillers, for insstance, when the user's description of a slot is ambiguous (see Section 7.2).

In summary, frames provide a natural representation for information about simple service domains: both *a priori* knowledge about the domain itself and information important to a system in interacting with the user to formulate a request for service. More specifically, the advantages of a frame-based representation for a simple service domain include:

- an easily manipulable representation for the overall task of interacting with a user to formulate a request for a simple service; the structuring of a frame into slots makes it easy to tell:
    - what parts of the request have been formulated,
    - what vital parts are missing,
    - and to what part the conversation is currently directed.
- convenient storage associated with each frame slot for information relevant to that slot including:
    - information about the type and structure of the entities supposed to fill the slot,
    - procedures to transmit fillers appropriately from one slot to another and provide special strategies to fill selected slots,
    - default fillers,
    - temporary information about partially described and alternative fillers.

In addition, frames have already been used successfully in systems that provide simple services.

## 5. Explanation Facility

A gracefully interacting system must be able to deal reasonably with any questions that its user sees fit to pose. In this section, we consider a limited set of question types that appear to us the most important for gracefully interacting systems, especially those operating in simple service domains.[1] The set includes questions about the system's abilities, its actions and the motives for them, other events within the system's experience, and hypothetical questions based on these types. We call the ability to answer questions of these types appropriately an explanation facility.

The set of types does not cover all reasonable questions, even for simple service domains, so that a system dealing only with these types could well find itself faced with legitimate questions that it cannot answer, or even comprehend. To deal with such cases, and with other cases of complete incomprehension, we also describe a technique, related to the explanation facility, by which a gracefully interacting system can extricate itself from situations in which its inability to comprehend has led to a totally confused dialogue.

### 5.1. Explanation Types

The explanation facility we propose deals with four different question types:

- Questions about ability:

  Can you give me a number in Tokyo?
  Can I make a reservation?
  How can I make a reservation?
  Can you give me a reservation for next Tuesday?
  What places can you give listings for?
  What can you do?

  Despite the fact that many questions ostensibly about ability are in reality requests to perform the embedded actions (with the service provider as the agent instead of "I" in appropriate cases), some ability questions must be answered literally. This requires a gracefully interacting system to have an explicit model of its own abilities and (some of) its inabilities.

- Questions about events:

  What did you just say?
  Why do you need to know my name?
  Have you made a reservation for Mr. Smith?
  Why can't you make the reservation for eight?
  Will you hold the reservation against late arrival?
  Why not?

---

[1] This impression is based on an analysis of some human interactions in such domains.

Dealing with such questions clearly requires a memory for what has already occurred in a conversation, together with knowledge about the goal structure of the conversation as described in Section 6.2.

- Hypothetical questions (embedding either of the two preceding types):

  Can you give me the number if I give you the address?
  Could you give me a reservation tonight if the party size was three?
  Will you hold the reservation if I put down a deposit?

  Answering such questions requires an ability to construct representations of the hypothetical conditions without getting them confused with the current situation or the user's previously stated preferences.

- Factual questions:

  Where are you located?
  Is there a charge for this service?
  What are your opening hours?

  These questions typically involve non-systematic domain-dependent information, and must be answered on an individual basis.

The repertoire of question types that we propose to deal with in our explanation facility is very far from a full spectrum of all the types of questions that can occur in ordinary human dialogue or discourse; this is clear from the range of question types considered in the work of Charniak [5], Lehnert [24], Scragg [35], and others. Nevertheless, we believe that such an explanation facility, together with the method for dealing with confused dialogues discussed below is sufficient for a gracefully interacting system in a simple service domain. In particular, the explanation facility proposed here is of much greater scope than that provided in more conventional question-answering systems such as LADDER [33], PLANES [40], LUNAR [46], etc. which have tended to concentrate on answering factual questions; the factual questions that such systems answer can, however, be much more complicated and are dealt with in a much more systematic way than we are proposing here.

In the remainder of this section, we will consider the various question types in more detail. The next two subsections will deal with ability questions, first in general human conversation, and then in restricted domains. The next two subsections deal with event and hypothetical questions respectively. Finally, we discuss a technique based on the explanation facility by which a gracefully interacting system can extricate itself from situations in which its inability to comprehend has created confusion.

### 5.2. Questions About Ability - Indirect Speech Acts

In general human dialogue, second person questions about abilities can be interpreted in two quite distinct ways. They can either be taken literally ("Can you swim?"; "Can you lift

this bar-bell?"), or be interpreted as requests for the listener to perform the action embedded in the question ("Can you open the window?"; "Can you tell me your address?"). The distinction between the two modes of interpretation does not depend on the question itself, but rather on the context (both linguistic and non-linguistic) in which it is spoken. The examples given above to illustrate the two modes of interpretation could all be interpreted the other way in a suitably chosen context; the intended interpretations are simply the more common ones.

The difference between the two modes of interpretation is accounted for by the linguistic theory of speech acts [1], [36]. In brief, the theory says that the listener will interpret an ability question as a request to perform the embedded action if he believes that the speaker already knows whether or not he is able to perform the embedded action; thus, "Can you open the window?" will generally be interpreted as a request, because, unless these are unusual circumstances, the listener will assume that the speaker believes that he is able to open the window. Using an ability question to express a request in this way is called an indirect speech act.[1]

The mode of interpretation of a second person question about ability is also correlated with the specificness of the question. A non-specific question such as ("Can you chop wood?") is much more likely to be interpreted as a literal question about abilities, than a more specific question ("Can you chop this wood?"). This distinction is somewhat related to the previously mentioned rule for choosing between the two modes of interpretation, in that if the speaker were unsure about a general ability of the listener, there would be no point in asking about a more specific ability.

One further point to note is that even though the listener interprets an ability question literally, the alternative mode of interpretation may also be relevant. In particular, if there is any possibility that the speaker may wish the listener to perform a possibly more specific instance of the action embedded in the question, then the listener is likely to suspect that the speaker actually does wish him to perform such an action.

A: Can you chop wood?
B: Yes, what do you want me to chop?

In this example, it is a moot point whether B interpreted A's question literally or not. The most important point is that B realized that A might want B to chop some particular wood, even though A's question was more general.

---

[1] Of course, this is not the only kind of indirect speech act. Requests for action, for instance, can also be expressed as questions about preferences ("Do you want to open the window?"), statements about preference ("I would like you to open the window."), and even more obscurely ("It's cold in here.")

### 5.3. Questions About Ability in a Restricted Domain

In this subsection, we consider what implications the human methods of dealing with ability questions have for a gracefully interacting system.

It is safe to assume that for the forseeable future any gracefully interacting system will operate in a highly restricted domain, and be able to provide its users with a highly restricted set of services. Making this assumption allows us greatly to simplify the treatment of direct questions by the user about the systems ability. In brief, if the user asks the system about its ability to do something that it can do, then it is reasonable to assume that the user is asking for the service to be performed; on the other hand, if the user asks about the system's ability to do something that it cannot do, a negative answer is appropriate. Thus, for a directory assistance service:

U: Can you tell me the number for Jim Smith?
S: Yes, it's 5629.

U: Can you connect me to Jim Smith?
S: No, I'm sorry! I Can't.

An interesting consequence of this strategy is that a system requires no explicit model of its abilities to answer ability questions which have positive answers. Since such questions are treated in the same way[1] as requests to perform the embedded action, the system need only recognize the embedded action and perform it. To do this, the system only needs to be able to recognize exactly those actions that it can actually perform. Clearly, this constitutes an ability model, but of an implicit kind.

The chief drawback to such an implicit model is that it treats all ability questions with negative answers in the same way, and furthermore, relies on non-recognition of the action embedded in the question. It is, of course, unrealistic to imagine that all the embedded actions that the system cannot do can be recognized, but recognition of some of the more likely ones would allow the system to help users with reasonable misconceptions about the system's ability, rather than just giving them a flat negative.

U: Can you connect me to Jim Smith?
S: No! You'll have to dial yourself, the number is 5629.

Another situation in which an explicit model of ability and/or inability is useful occurs when a service has parameters, and the system can only perform the service for certain values of

---

[1] The treatment cannot be completely identical since "Yes, it's 5629" is not an appropriate response to "What is the number for Fred Smith?", but apart from the format of the response there is no apparent need to discriminate.

the parameters.

U: Can you given me a number in Rome?
S: No, I have only local numbers.

U: Can you make me a reservation for next month?
S: No, we only accept reservations a week in advance.

The second example particularly shows the importance of telling the user the limitations the system places on the parameter; if it just replies in the negative, the user has no way of knowing what prevents his request from being fulfilled, and consequently no way of deciding whether he can modify his request so that the system can help him. Note also that this strategy applies not only to requests phrased as ability questions, but also to requests in any other format.

U: I would like a reservation for next month.
S: I'm sorry, we only accept reservations a week in advance.

In discussing questions about ability in human dialogues, we observed that the more vague an ability question, the less likely it was to be interpreted as a request for action. It seems that a gracefully interacting system need not make this distinction. All ability questions with positive answers can be treated as requests for action, while all with negative answers should be answered negatively, giving whatever extra information is appropriate as discussed above.

U: Can you give me a number?
S: Yes, whose number would you like?

U: Can you make me a reservation?
S: Yes, what time would you like?

As these examples suggest, the requests implied by vaguer ability questions are typically too vague to be fulfilled directly. In other words, the vagueness takes the form of the absence of (or vagueness in) certain parameters of the requested task, without which the task is not well-defined. In the examples above, the user does not specify the name of the person for whom the number is required or the time or other details of the reservation. Appropriate action for the system is not to reject the request because it is incompletely specified, but rather to ask for specifying information, as in the examples above. Of course, the extra information the user goes on to supply may specify a request that cannot be fulfilled, but such problems must be dealt with as they arise. Again, note that formulation of the request as a question about ability has little to do with the system's response; vague requests formulated as other question types, statements, or imperatives should be dealt with in essentially the same way.

There are exceptions to our proposal for handling all ability questions as requests for action. First, "wh"-questions about ability ("Where can you give listings for?") should generally be answered directly; although if the answer is awkward because of its size or some other reason, it may still be appropriate to treat the question as a request for action ("Where do you want a listing for?"). Secondly, the extremeness of vagueness in which the embedded action is "do" ("Can you do something for me?", or more naturally in "wh"-form, "What can you do?") should cause ability questions to be treated literally, and make the system give an accounting of its abilities. Clearly, both these exceptions require a system to have an explicit model of its own abilities. They correspond closely to calls on a "help facility" of the type commonly found in current (non-graceful) interactive systems.

To end this section, we might note two points: first, virtually all that has been said above about second-person ability questions applies equally to many first-person ability questions ("What can I do?", "Can I make a reservation for 7 PM?") in which the user places himself in the active role. In fact, the only cases in which the transformation is not applicable are those in which the user was a recipient in the second-person version ("Can you tell me the number for Jim Smith?"). In general, then, first-person ability questions should be treated exactly the same as the corresponding second-person questions. Secondly, "how" questions ("How can I make a reservation?", or even "How do I make a reservation?") can usually be treated in the same way as the corresponding ability question without the how, i.e. as requests to perform the action specified. "How" questions for which this transformation does not produce a request fulfillable by the system ("How can I get there?", "How do you get any customers at those prices?") are either informational questions which should be recognized separately (see Section 5.1), or are questions that a gracefully interacting simple service system could not be expected to answer satisfactorily.

### 5.4. Event Questions

In this section, we turn from ability questions to questions about events, covering events both in the past and in the future, and the motives behind them. As in the case of ability questions, the restricted domains of forseeable gracefully interacting systems make event questions much more tractable. The only event questions to which a gracefully interacting system can be expected to give an informed reply are those concerning its own actions and their results and motives, along with those actions of its users which it can observe directly. Questions about events outside its direct experience will be incomprehensible.

Examples of questions that should be dealt with include:

What did you just say?
Did you just ask for my name?

Why do you want to know my name?
Why won't you make the reservation for 8pm?

and using examples from an electronic mail domain:

Has the message I sent yesterday been delivered successfully?
Will this message be delivered immediately?

The first two examples are explicit indications of incomprehension as discussed in Section 2.3. Such questions about what has just been said serve a robust communication function and should therefore be treated as special cases. Answering other event questions requires a history of all the events that have taken place in the current interaction, plus in the case of the system's actions, the reasons for them. The SHRDLU system of Winograd [44] was the first to maintain such a history of events and their reasons. As the penultimate example shows, event questions can refer to events outside the current interactive session. Presumably, it is impractical to maintain a detailed history of all previous interactions, and instead sufficient to remember only "major" events, such as the delivery of a message in the case of an electronic mail system. Questions to the system about its future actions, as in the last example, require the system to "imagine" that the future has arrived under prescribed conditions and to observe the results; such questions are thus closely related to the hypothetical questions discussed in the next section.

Before leaving this section, we should note that, like ability questions, event questions can also be indirect speech acts. For most yes-no questions, a bare negative answer is generally not an acceptable response, since such questions normally involve an indirect speech act asking for an explanation in the negative case. Thus, for "Was the message delivered successfully?", it is unacceptable merely to reply "No!"; the reason for non-delivery should also be given. Again, "Will the message be delivered immediately?" not only seeks information, but would normally be interpreted as an indication that the user wishes the message to be delivered immediately. As a third and final example, the user of a restaurant reservations service says in the middle of his conversation "Did I say there would be eight in the party?", he is probably not even interested in the direct answer to the question, but only in making sure that the system knows there will be eight in the party. The recognition of such speech acts is a difficult problem which has been addressed in more general terms in the work of Cohen [7], Levin and Moore [25], Lehnert [24], and others. Adapting such work to the requirements of graceful interaction, and using all the constraints that simple service domains give is an important but unsolved problem.

### 5.5. Hypothetical Questions

Besides being able to answer questions in the context of the current situation, a gracefully interacting system must also be able to deal with questions based on a hypothetical context

invented by the user. As usual, dealing with such questions in limited domains is very much easier than for general human conversation, simply because the number of variables about which hypotheses can be made is so very restricted.

In the case of simple service systems, the most important class of hypotheses are those concerning the value of one of the slots of the service or one of the subslots of a slot. A user may ask an ability question based on the hypothesis that such a slot has a certain value, or even that the (unspecified) value of the slot is known to the system. Examples include:

Can you give me a reservation if the party-size is three?
Can you give me the number, if I give you the address?

U: I'd like a reservation for this evening.
S: I'm afraid we're fully booked.
U: Could you give me a reservation for tomorrow?

To answer such questions, a gracefully interacting system must first construct a representation of the hypothesized situation, and then evaluate and answer the question. In doing this, the system must be careful not to confuse the representation of the hypothetical situation with any other conflicting situations the user has previously specified, as in the last example. The user may wish to return to his previous specification, or even to introduce others, and switch between them several times (this might be quite common for, say, an airline reservation system). In other words, the system must have a way of representing alternative situations independently, of sharing common information between them, and of switching back and forth between the alternatives. Such ideas have been explored in knowledge representation systems such as CONNIVER [38] and NETL [10], and in work by Hendrix [21], Hayes [16], and others. The methods thus developed are suitable for use with the frames type of representation proposed in Section 4.3.

Even after a representation for the hypothesis has been constructed, it is not always possible to treat a hypothetical question in the same way as its embedded question would be treated if it were posed in a context corresponding to the situation hypothesized. In particular, the indirect speech act may be different. Thus "Can you give me a reservation for eight pm tonight?", spoken in a context in which the party size has been previously established would normally be interpreted as a directive to make a reservation at the stated time if that is possible. On the other hand, "Can you give me a reservation for 8pm tonight, if the party size is three?" would normally be interpreted as an exploration of possibilities, rather than a request for action, and might well have been preceded by a discussion in which the party-size was different; it should therefore be answered literally. However, once it has answered yes or no, the system should not forget the hypothesis, since the user may then decide to make a reservation based on that hypothesis.

Finally, note that hypotheses can be made about things other than slot values, as in: "If I had called two hours ago would you have been able to give me a reservation?". Answering such a question would require an ability to change the context by restoring the reservations data-base to its state two hours ago. For a practical system, one would have to balance the utility of answering such questions against the expense of facilities for changing the context in the way required.

### 5.6. Ability Models as a Safety Net

In the preceding sections we have discussed what is essentially a passive explanation facility: one that issues explanations in response to a user's express desires (whether those desires are expressed literally, or in an indirect speech act). In this section we discuss a more active type of explanation facility: one which volunteers explanations of its own abilities in circumstances in which the user might benefit from such explanations. We only propose one such circumstance here - when the system has repeatedly failed to understand the user and the conversation is in danger of breaking down completely. For a more sophisticated system there might be others, such as when the user appeared to be using a more complicated strategy to accomplish his goals than was really necessary.

We claim then that, when a gracefully interacting system has repeatedly failed to understand its user, it should use its ability model to let the user know what it can do for him, plus possibly what it needs to know to do it. There are two benefits: first, it allows the user to see whether the system can help him in what he wants to do; and secondly, if the system can help, the way in which it states its abilities suggests to the user a phrasing for his requests that the system would be able to understand.

U: What is the number for Jim Smith?
S: Jim Smith's number is 2597.
U: Thank you. Do you know how the weather is where you are?
S: Are you asking me for another number?
U: No! I'm asking you about the weather.
S: I'm sorry! All I can do is give you telephone numbers. I am completely ignorant of everything else.

In this example, the system cannot understand the user's questions about the weather. The first time he asks, it asks a question based on what it expected him to say (this strategy was discussed in section 2.3), but this does not produce anything more understandable from the user. The system then assumes that the conversation is breaking down and uses its ability model to inform the user of what it can do for him (and intensionally what it cannot). At this point, the user is fully aware of why the system is not cooperating more directly with him by fulfilling his request for information about the weather, and so is much less likely to become

frustrated than if he didn't understand the reason for the problem.

The following example shows how the way the system phrases its statement of abilities can aid the user in formulating his request.

U: I want to come and eat at your restaurant.
S: I'm sorry! All I can do is make reservations. I am completely ignorant about everything else.

We are assuming that the system cannot understand the user's request because of the unusual wording, and that this fragment was preceded by other unsuccessful attempts at communication, so that the system has fallen back on its ability model. Now, if the user cares to heed the system's statement of its ability, he might try something like: "Then please make me a reservation.", which the system should be able to understand. Of course, this feature implies that the system can comprehend any appropriately transformed version of its own statement of abilities (see Section 8).

Finally, note that this use of an ability model, while vitally important for a computer system of limited understanding, has little basis in real human dialogues. The reason for this is clear: humans understand each other so well and have such a breadth of knowledge that conversations hardly ever break down into repeated incomprehensions; so humans do not have any real need for a last resort strategy of the kind we have discussed.

### 5.7. Summary of Explanation Facilities

We can summarize this section as follows:

- A gracefully interacting question should be able to answer several different types of question from its user, including questions about its abilities, about its actions and the motives for them, about other events in its past experience, and hypothetical questions based on all these other types.

- In human dialogue, second-person questions about ability are interpreted in one of two distinct modes: either literally or as requests to perform the action embedded in them.

- In general, a gracefully interacting system need not make this distinction; second-person ability questions with positive answers can be treated as requests for action, and those with negative answers should be answered negatively. This does not require an explicit ability model.

- However, an explicit model is useful for the negative cases when it is only incorrect parameters that prevent the system from performing the embedded action, and for giving useful information besides a flat negative in other commonly occurring negative cases.

- Other cases in which ability models are important are "wh-" ability questions, and

very vague ability questions in which the embedded action is "do."

- Most first-person ability questions can be treated in exactly the same way as the corresponding second-person questions.

- To answer event questions, including those about its own actions and the motives for them, a system needs to maintain a history of all events that have taken place in the current interaction, plus major events from previous interactions, together with its representation of the goal structure of the conversation at the time of those events.

- Event questions, especially those of a yes/no type can also be indirect speech acts.

- Answering hypothetical questions requires an ability to construct and swop between temporary contexts which represent the conditional parts of the hypothetical questions; the range of aspects of the context which can be manipulated in this way restrict the range of hypothetical questions which can be dealt with.

- A gracefully interacting system can also use its model of its own abilities to extricate itself from totally confused dialogues by telling the user what it can and cannot do for him.

## 6. Goals and Focus

In this section we deal with the intimately related topics of the goals of the participants in a conversation and what their attention is focused on at any given point in the conversation. Both these items are very complicated in general human conversation and a thorough treatment is far beyond the scope of this paper. Instead, we will largely confine ourselves to dealing with these phenomena in gracefully interacting simple service systems. We will find that high-level goals can be treated very simply for such systems, since there are very few goals that the system can recognize in the user (essentially only those to make use or find out about the services provided by the system, plus an undistinguished set of others that the system cannot help with), and since the system has no goals except to help the user satisfy his. Subgoals that arise during the satisfaction of the top-level goals must, however, be treated more generally in order for a system to understand what is happening in a conversation.

The participants in a conversation share a common view of what the conversation is about, which we call the focus of the conversation. This shared focus allows them to economize in what they say through anaphoric reference and ellipsis. It is thus very important for a gracefully interacting system to be able to follow the shifting focus of a conversation in order to interpret its users' ellipsis and anaphora. Goals and focus are clearly highly related in any case, but for simple service systems we find it expedient to equate them. While other

definitions of focus have aided in the resolution of anaphora, the view we propose aids also in the treatment of ellipsis. We will describe how this view of focus helps in the resolution of ellipsis and anaphora, and discuss strategies for keeping track of it.

### 6.1. Goals

Human conversation is very goal-oriented. In other words, virtually everything a person says during a conversation is intended to achieve some definite aim. An ability to determine the goals of its user is thus of prime importance for a gracefully interacting system.

In general human conversation, the types of goals that can be pursued are many and varied. They range from highly specific (trying to find someone's telephone number) to extremely vague (maintaining "interesting" small-talk), from quite straightforward (trying to find out someone's origin by a direct question) to downright devious (trying to manoeuvre someone into pronouncing a certain word so that you will know his origin); from completely cooperative (obtaining a number from a helpful directory assistance operator) to totally antagonistic (a cross-examination at a trial). Fortunately, as we shall see, a gracefully interacting simple service system need concern itself with only a highly restricted subset of these various types of goals.

Not surprisingly, the goals people have affect what they say and the way they say it. Even more importantly from our point of view, the way a listener interprets what a speaker says is dependent upon the listener's view of the speaker's goals. The way in which this happens is the subject of the linguistic theory of speech acts by Austin [1] and Searle [36], and has also been studied in more computationally oriented work by Cohen [7] and Mann, Moore, and Levin [27].

The problems involved in dealing computationally with the full range of goals and speech acts are immense. In both pieces of work cited, it was found necessary to maintain a model of the beliefs of both participants in a conversation, including beliefs about the beliefs of the other participant. In addition, Levin and Moore [25] found it useful to recognize a number of distinct patterns of expressed goals and characteristic responses; examples included asking for information, requesting help with a problem, and requesting a service.

While the work just mentioned has explored the problem, many unresolved difficulties remain in the recognition and modelling of all the types and patterns of goals that can arise in general conversation. So it is fortunate that graceful interaction, at least for simple service domains, involves the recognition and modelling of only a highly restricted set of goal types. For the remainder of the section we will concentrate our attention on this restricted problem.

A gracefully interacting simple service system can make the following two sweeping simplifications in its modelling of goals:

- it has no independent goals of its own; its only goal is to help the user fulfill his goals;
- the user's goals are either to avail himself of the system's highly limited services, or fall into an undistinguished class for which the system is unable to help the user.

Note that these simplifications apply only to primary or top-level goals; lower-level goals or subgoals which can arise during an attempt to satisfy a top-level goal must be treated more generally as we will see in the next section.

The two restrictions require little justification. The first follows from the notion of a gracefully interacting system whose reason for existence is to serve its user. Such a system would have no interests of its own to worry about, as does every human participant in a conversation, and so it need have no goals other than those it can recognize in its user. In attempting to satisfy its users' goals, it can, of course, generate subgoals which cause it to take the initiative in the interaction, but such initiatives are always subservient to the user's top-level goal.

The second restriction is appropriate because it is futile for a gracefully interacting system to recognize goals in the user that it cannot help to fulfil. The only goals that a gracefully interacting system needs to recognize in its user are those within its domain of expertise, plus perhaps a few other closely related areas that it explicitly knows it cannot fulfil, but for which it can offer some helpful advice (see Section 5.3). Without becoming significantly less graceful, it can treat all other goals that the user might express with undifferentiated incomprehension. For instance, it would be of little use for a directory assistance program to recognize that its user was trying to find out the state of the weather, or trying to ask it out to dinner, since it would be quite unable to fulfil either of those goals of the user.

Neither of these simplifications is generally true of gracefully interacting systems outside the simple service class. A graceful supervisory or instructional program might, for instance, have to choose between cooperating in fulfilling a user's expressed goal, which it knows to be a bad goal to have, and correcting the user. Again, the range of goals that a user might reasonably express to a secretarial system could be quite large.

In summary, the combination of recognizing a highly limited number of goals in the user, and making the user's goals the system's own, allows the treatment of high-level goals for simple service domains to be extremely simple. The system need have no high-level goals of

its own; it just cooperates with the user in fulfilling his goals. Moreover, the system can assume that the goals of the user are either to use (and/or perhaps find out about) the service it offers, or are outside the competence of the system. Thus, graceful interaction in a simple service domain does not require a sophisticated model of its own and the user's goals and motivations, such as would be necessary for a more general interaction.

### 6.2. Subgoals In Simple Service Domains

While top-level goals can be treated in a very simplified way by a gracefully interacting simple service system, the subgoals that can arise during fulfillment of the top-level goals must be treated much more generally. As we will see, subgoals for simple service systems can be nested arbitrarily deeply, can originate from the system as well as the user, and involve the acquisition or imparting of specific pieces of information or descriptions.

Subgoals arise because top-level goals often cannot be accomplished in a single step. Thus a user may be unable or unwilling to specify all the components of, say, a restaurant reservation (time, party size, etc.) in a single utterance. Instead, he may spread the specifications over several different utterances; we can say that in each of those utterances he is pursuing the subgoals of imparting the specifications of the corresponding components.

Unlike top-level goals, subgoals can originate from the system as well as the user. A user may fail to volunteer specifications of certain parameters of the service, or may specify parameters in an ambiguous or unsatisfiable way (see Section 7). In such cases, the system may take the initiative by formulating and pursuing a subgoal of trying to acquire the appropriate information from the user. If the user of a restaurant reservations system, for instance, does not volunteer the size of his party, the system will have to ask him explicitly. Such system generated subgoals are, nevertheless, derived from and subordinate to the goals of the user, and should not be pursued blindly if the goals of the user change.

There can be several levels of subgoals; that is, subgoals can themselves have subgoals. So far, all our examples have been in terms of the primary parameters of the service offered by a simple service system: subgoals on the part of the user to specify a parameter to the system, and subgoals on the part of the system to obtain the specification of a parameter from the user. In terms of the frames representation described in Section 4.3, the subgoals have been to fill the slots of the frame representing the service. However, these subgoals may themselves not always be achieved in one step. If the system, for whatever reason, does not understand all of a description of a parameter (slot) by the user, it can set up the subgoals of obtaining the missing part of the description from the user.

U: I would like the number for Jim <GARBLE>.
S: Jim who?

In this example (see Section 2.3), the user is pursuing the goal of trying to find a telephone number, including the subgoal of trying to identify to the system the person for whom he wants the number. This subgoal succeeds only in part, because the system cannot understand the surname, and so the system sets up a sub-subgoal by explicitly asking the user to redescribe the surname. Note that the subgoal generated by the system is subordinate to the user's original subgoal. Other common examples of subgoals of subgoals occur when a user's description of a slot is ambiguous or unsatisfiable (see Section 7) and the system attempts to resolve the difficulty.

If a goal can have several subgoals which can in turn have subgoals of their own and so on, trees of subgoals of the type common in planning systems (see Hayes [17] and Sacerdoti [34], for example) can be produced. However, since a simple service system does not need to plan an entire conversation out in advance, there is no need, as is usual in planning systems, to generate complete trees of subgoals in advance (though see Section 6.4); indeed, an interaction in which the system insisted that all parameters were specified in a fixed order (and the generation of such a tree would imply this) would be quite ungraceful. More important is the dependency between subgoals and their parent goals. Since subgoals are generated in an attempt to fulfil higher-level goals, they must be abandoned rather than blindly pursued if the parent goals are abandoned or changed.

U: I would like the number for Mr. Smith.
S: Do you mean Jim Smith or Joe Smith?
U: I'm sorry, I mean Fred Jones.

In this example, the system generates a subgoal of deciding between two alternative fillers corresponding to the ambiguous slot specification by the user. However, the user does not cooperate with the system's subgoal, but instead changes his own subgoal from which the system's subgoal is derived. This means that the system should now abandon its subgoal and attempt to fulfil the user's revised subgoal. Note that the commitment of the system always to cooperate without any corresponding commitment on the part of the user means that such changes in goals will always originate with the user.

The dependency of subgoals on the goals that generate them is not the only form of dependency that need concern simple service systems. Since parameters to services can be interdependent, subgoals involved in establishing one slot can be dependent on another slot being filled or remaining unchanged. For example, the time, date, and party-size of a restaurant reservation are mutually constraining, so that changing one may require changing another. Work on representation for these more general types of dependencies has been done by Hayes [17] and Doyle [9].

To end this subsection, we might comment on the general form of subgoals that arise for

simple service systems. The most important are subgoals to impart or acquire information; commonly, the user will have subgoals of imparting the specifications for the parameters (slots) of the service, and the system will, when appropriate, generate subgoals for acquiring such specifications from the user. Lower-level subgoals of this type concern parts of these specifications, i.e. slots of frames which fill the slots in the main frame for the service (e.g. the surname for a name).

Subgoals can also concern the acquisition of information relevant and prerequisite to specifying a slot. Thus, prior to specifying the time of a restaurant reservation, a user might ask what time the restaurant closed. If the user was asking the question because he wanted a reservation as late as possible, this question would represent a subgoal of the subgoal of specifying the time. Since an informational question could also signal a switch to a completely different subgoal, knowledge of what information was relevant to which slot specifications would be useful to the system to enable it to keep track of which subgoals are current.

S: What time would you like your reservation?
U: What time do you close?

In this example it would be desirable for the system to appreciate that the user was probably cooperating with the subgoal it had generated, rather than introducing a distinct subgoal of his own. However, in general, arbitrary amounts of knowledge may be required to make this determination; the user might, for instance, ask how far the restaurant was from the Opera House in trying to specify the time of his reservation. The best strategy may be to recognize common examples of this type of subgoal as special cases, and answer all other informational questions with the presumption that they represent subgoals of the current goal or subgoal. Note that from this perspective, general questions about the system's ability can be viewed as subgoals of the user's top-level goal to make use of the system's services. Other methods of keeping track of goals and subgoals are discussed in Section 6.4.

In summary, subgoals can arise for both the user and the system and they can be nested several levels deep. The system should always cooperate with the subgoals established by the user, but the user cannot be expected always to cooperate with subgoals established by the system; he may pursue other subgoals independently, or change previously established goals or subgoals on which the system's subgoal depended. In either case, the system should follow the user's initiative rather than doggedly pursuing its own subgoals. The most common form of subgoal concerns the specification of a parameter of the service or of a sub-slot of such a parameter. Another common type seeks information useful for establishing such a specification.

### 6.3. Focus

When people engage in conversation, their attention is generally directed to a highly specific topic or focus. This focus of attention is typically the same for all the participants, giving rise to the impression that they are "talking about the same thing". A shared focus of attention allows dialogue participants to economize substantially on what they say through use of ellipsis and anaphora; they assume listeners can fill in the missing information by use of the common focus. Since people make such economies of expression very frequently and naturally, any gracefully interacting system must be able to keep track of the shifting focus of its conversation and use it to resolve the ellipsis and anaphora of its user (as well as perhaps generating its own). The problems involved in keeping track of conversational focus are the topic of the next section; in the remainder of this section, we will provide an expedient definition of focus for simple service domains, and show how such a focus can help resolve anaphora and ellipsis.

A definition of focus that is both precise and general is hard to formulate. Even though the notion of focus (or topic) has been used in computational systems by Grosz [14] and Sidner [37] as well as in more traditional linguistic work, including some by Grimes [13] and Hockett [22], it has been viewed in different ways according to the research goals involved. Grosz, for instance, modelled dialogues in which an expert guided an apprentice through a mechanical assembly task, and found it convenient to equate the focus of the dialogue with the set of objects involved in the assembly step currently being undertaken or under discussion. Sidner, on the other hand, dealt with monologues specifying arrangements for a meeting, and so took focus to be a complete single entity: either a meeting as a whole or some parameter of it, such as its time or place. In both cases, the entity or entities in focus are used to help resolve anaphoric references by providing a set of potential referents for such references.

Nor will we attempt to define focus in general, or even for all of graceful interaction; instead we will again confine ourselves to simple service systems. For such systems, however, we propose an extension of the notion of focus by equating it with the currently active subgoal. This view of focus subsumes the notion of a collection of entities to which attention is currently directed, since the current subgoal also defines a set of entities which are currently receiving attention - those involved in the subgoal. This set can be used in the same way as just mentioned to provide potential referents for anaphora.

Using the current subgoal as the focus also, however, helps in the resolution of ellipsis. The ellipsed information can often be filled in, via the assumption that the elliptical utterance is intended to fulfil the current subgoal, thus providing a complete interpretation for the

elliptical utterance. For instance, an answer of "X" in response to a question of "Do you mean X or Y?" could be seen as fulfilling the subgoal set up by the question of choosing between X and Y. Interpreted in the light of such a focused subgoal, the elliptical utterance, "X", has the coherent interpretation of choosing X instead of Y.

In other words, this extended notion of focus is useful in the same way as the more usual one in the resolution of anaphora, but unlike the other one, can also be used in the resolution of ellipsis. This latter use of focus appears to be novel, but see also the work of Levin and Moore [25]. Of course, it could be argued that the two aspects of focus - as a collection of potential anaphoric referents, and as a goal with the potential for providing ellipsed information - are separate and should not be dealt with in related ways. We believe, however, that the two aspects are sufficiently intertwined for it to be profitable to treat them through a common mechanism.

At this point, some concrete examples will make it clearer how focus, as we are proposing it, can help with anaphora and ellipsis.

U: I'd like the number for Mr. Smith.
S: Do you mean Bill Smith or George Smith?
U: I mean Bill.

Here the new user refers anaphorically to Bill Smith by use of the first name. This reference is easy to resolve, however, because the focus is on the user generated subgoal of filling the person slot of the directory assistance frame, and more particularly on the system generated subgoal of getting the user to distinguish between two alternatives; this subgoal involves two entities and the reference can easily be resolved to one of them. A similar resolution could be made if the user had, for instance, said, "I mean the first one." Note that this resolution depends upon the fact that the user's utterance does not change the current subgoal and hence the current focus. If we further change the example so that the user's reply is just "Bill", or "The first one", this ellipsis can easily be resolved by assuming that the user is cooperating in the focused subgoal of choosing between the alternatives, so that the object mentioned actually represents the user's choice. Note that there is no need to reconstruct a phrase such as "I mean Bill", (or "I think it's Bill, "Bill is the one I want the number for", etc.); it is enough to interpret the referent as fulfilling the system's subgoal of getting the user to indicate a choice (see Section 3.2).

Consider also the following extract from the extended example of Section 9.2, in which the time of a restaurant reservation is under discussion.

S: I can't give you eight o'clock; it would have to be seven or after nine thirty.
U: You don't have anything between those times?
S: No! I'm afraid we don't.
U: Well, the later time then.

The system's "it" in the first line refers to the time of the reservation rather than eight o'clock; this corresponds to the focus of filling the time slot. The system's utterance refines the goal of filling this slot into a subgoal involving a choice on the part of the user, and "those times" in the user's reply has to be interpreted in this context, since in particular, the referenced set of times does not include eight o'clock. The user's "anything" is less straightforward, however; if a referent had to be found for it, a non-specific reference to a time would probably be best, but it is probably preferable to interpret the utterance as a whole as asking the system to confirm its restriction on times, without trying to interpret "anything" individually. It thus sets up a subgoal on the part of the user, in terms of which the system's elliptical reply is easy to interpret. However, the focus on the choice established in the system's first utterance is still available, and must be used to interpret the ellipsis and anaphora in the user's second utterance.

The importance of focus, then, is related to the commonness of anaphora and ellipsis in natural language use, and they are very common indeed. Both allow a speaker to economize on what he says by omitting unnecessary information that his listener can fill in because their attention is focused on the same common ground. Since the user of a gracefully interacting system will naturally economize in this way, it is vital for the system to be able to use its awareness of the conversational focus to make good the omitted information.

In summary, for simple service systems it is convenient to equate the focus of the conversation with the subgoal that the user and the system are currently cooperating on. Anaphoric references can often be resolved into objects involved in the subgoal currently in focus, and ellipsis can be resolved by assuming that the elliptical utterance is fulfilling the focused subgoal.

### 6.4. Keeping track of focus

Since focus is so important in the interpretation of ellipsis and anaphora, and since these phenomena are so commonplace, it is very important for a gracefully interacting system to keep track of the focus as it shifts during the course of a conversation. It is easy enough for a system to tell when the focus changes as a result of some new subgoal it establishes, but what about when the user changes subgoals and therefore focus?

The problem is a very difficult one in general, and even for simple service systems, we can

only suggest a direction for experimentation. The main problems are the unpredictability of focus shifts in terms of both when they occur and what the new focus will be. As Sidner [37] observes, focus shifts cannot be predicted in advance, they can only be detected after they occur. In the remainder of this section, we will investigate what the new focus can be after a shift (the answer will be virtually anything), and suggest some approaches to detecting such shifts.

A very common pattern of focus shift arises when goals are refined into subgoals, which in turn spawn subgoals of their own. For example, in:

U: What is the number for Bill Smith?
S: Did you say Bill or Phil?
U: Bill with a 'B'.
S: 'B'. O.K. The number is 2597.

the user first focuses on the person parameter (slot) of the directory task (frame), the system then changes the focus to the first name of the person and in particular to a choice between two alternatives for that slot. The user then cooperates with the system's goal and chooses one of the alternatives, but in doing so refines the focus yet again to the first letter of the first name. In its reply, the system first maintains the focus by acknowledging the letter, and then shifts the focus drastically by providing the user with the number he desired.

The last focus shift can be thought of as popping the nested set of foci that had been built up through the preceding utterances, and then moving on to the next task - in this case performing the requested service since its parameters are now complete. Popping back through all the nested levels and going on to the next major subgoal is probably the most common way of terminating such nesting, but partial popping can also occur:

U: What is the number for Mr. Smith?
S: Do you mean Bill Smith or Joe Smith?
U: I mean Jim Smythe.

Here the user does not cooperate with the system's subgoal of distinguishing between two alternatives, but instead pops back to the previous level and retries his goal of specifying the person slot to the system.

Focus shifts do not, however, always follow this nice stack discipline. For instance, the user can at any time reopen a focus that was apparently closed, either as a result of something the system tells him which might make such a shift more predictable, as in:

U: I would like a reservation for two tonight.
S: What time would that be for?
U: About seven.
S: I'm afraid the earliest I could give you is eight thirty.
U: What about tomorrow night?

or for reasons best known to himself:

U: I would like a reservation for a party of six tonight.
S: What time would that be for?
U: No! There will be seven of us.

Even worse, the user can shift the focus to subgoals of previously open foci, which themselves have never been open, as in:

U: What is the number for Bill Smith?
S: The number for Bill Smith is 2597.
U: No! I said Phil with a 'P' as in parrot.

where the user not only reopens the focus on specifying the person, but jumps immediately into specifying the first name slot. Note that the specification for the surname remains the same, which is added evidence that the old focus of filling the person slot is reopened and then refined.

These examples suggest that at any time there is a tree of potential foci, i.e. subgoals which may become the current focus; for a simple service system, the root of the tree would be the top-level goal of getting the service performed, the next level would be the goals of specifying the main parameters or slots of the service, the next level, specifying their slots, etc. The stack-like pattern of focus shifts, described above as the most common type, then corresponds to depth-first shifts down one branch of the tree, followed by a return up the branch and a transfer to the highest node of another branch. Reopening of old foci corresponds to visiting a node in the tree more than once, and shifting focus to a subgoal of a previously open focus, which itself has never been open, corresponds to visiting for the first time a node whose parent has been visited before.

In the examples above, the shifts across branches always involved shifts to completely different branches, since they involved changes in which top-level parameter of the service was being considered. Shifts can also occur across branches which meet below the root of the tree, as the following example shows.

U: What is the number for Mr. Smith?
S: Do you mean Joe Smith or Fred Smith?
U: I mean the Smith on the eighth floor.

Here the user does not cooperate with the system's subgoal of distinguishing between the Smiths by first name, but instead chooses to distinguish them by location, a sibling subgoal of the goal of filling the person slot.

Finally, a user can always completely abandon his top-level goal, and thus move the focus entirely out of the tree, as in:

U: I would like a reservation for two this evening.
S: It would have to be after ten.
U: I'll try somewhere else.

Note also that in this example the user covers two subgoals in his first utterance; the system must be prepared to deal with problems in one of them without forgetting the other.

The net conclusion from all these examples is that while there are certain common types of focus shift, which follow a stack discipline corresponding to progress through the specification of the service and the several levels of detail that can be involved, focus can shift to virtually any part of the tree of potential foci, including shifts to old, apparently closed, foci or to subparts of such old foci that have never themselves been open.

Previous work on keeping track of focus has been based on more restrictive assumptions. Modulo their different views of focus, both Grosz and Sidner have assumed that focus shifts in a quite orderly way through a tree of potential foci; in particular, they assumed that no node or branch of the tree was visited more than once, and in some cases, that the branches of the tree were traversed in a partially fixed order. They then detected focus shift from mention of entities that belong in foci either further down or further up the same branch of the tree as the current focus, or in a focus on one of the permissible next branches. We have no immediate solution for the detection of more general types of focus shift; a good starting point might be, in the same way as Grosz and Sidner, to search for appropriate foci when the entities or goals mentioned do not fit in the current focus, but to search over the entire tree of possible foci for the task domain. In addition, since old foci can be reopened and specific objects associated with them reused, the relevant information must be associated with the nodes that have already been traversed.

### 6.5. Summary of Goals and Focus

We can summarize the main points made in this section as follows:

- Ordinary human conversation is highly goal oriented; people almost always are pursuing one or more goals in saying what they say.

- A gracefully interacting simple service system need have no high-level goals of its own; it merely cooperates with the user in fulfilling his.

- The only goals such a system need recognize in its user are those it can help satisfy; this precludes the need for a sophisticated model of top-level goals.

- Subgoals must be treated with greater care; they can be nested arbitrarily deeply and originate from both the system and the user, but the system generated subgoals are always derived from and subservient to those of the user.

- Common types of subgoal concern the specification of parameters of the service, or one of their subslots; others seek information useful for establishing such specifications.

- People use a shared view of what a conversation is about, called the focus of the conversation, to economize substantially on what they say through anaphora and ellipsis.

- For a simple service system, viewing focus as the currently active subgoal helps in the resolution of both anaphora and ellipsis.

- Although there are some common patterns of focus shifts, the focus of a conversation has the potential to shift unpredictably to any of a tree of potential foci.

# 7. Identification from Descriptions

## 7.1. The Identification Problem

A fundamental component of human conversation is the ability of a listener to use a speaker's description of a previously memorized entity (object or event) to identify and recall the entity. If a definite focus of attention has already been established in the conversation, then as we saw in Section 6.3, descriptions can be quite cryptic or anaphoric (the woman; the second one; it). On the other hand, if no context has been established, or if the object to be described is not in the established context, much fuller forms of descriptions must be used (the camping trip we went on last Easter; a word beginning with 'O' meaning fawning or servile; the lady that I saw you with last night). One of the more remarkable features of human memory is the ability to identify from a suitable description virtually any previously experienced or known object or event (that man we met at the second hotel we stayed at when we went on vacation last year).

When he describes an entity, a speaker aims to identify the entity uniquely to his listener. To do this, he uses his beliefs about the listener's state of knowledge to construct a description that is informative enough to identify the object uniquely to the listener (possibly with respect to a current context). Most usually he succeeds, but sometimes a listener will recall more than one entity fitting the description, or may not be able to identify any entity filling the description. In other words, there are three possible outcomes to any attempt to identify an entity through a description:

unique - only one entity fits the description
ambiguous - more than one entity fits the description
unsatisfiable - no entity fits the description

In accordance with the Principle of Implicit Confirmation (Section 2.2), a speaker assumes that his communication is successful, and in particular that his descriptions identify unique entities for his listener, unless the listener indicates otherwise. The listener thus need do nothing if a description falls into the unique case, but must indicate a problem to the original speaker in the ambiguous or unsatisfiable cases. In indicating such a problem, the listener will usually initiate a dialogue during which the description is revised to specify a unique entity to the listener.

Identification from descriptions is clearly very important for a gracefully interactions system which must identify entities from its user's descriptions, and must in turn describe things to its user. A directory assistance must be able to accept descriptions of its listings, and in turn describe the corresponding numbers to its user; and a restaurant reservation system must be able to accept and give out descriptions of times, dates, party sizes, etc..

Fortunately, identification from descriptions for gracefully interacting systems, does not involve the quite awe-inspiring range of description and recall found in human conversation. In simple service systems, in particular, entities that the system must recognize from the user's descriptions are essentially the entities that serve as parameters to the service, and these typically fall into a few highly restricted categories such as listings for directory assistance, times, dates, party sizes, and names for restaurant reservations, and mailboxes, hosts, and messages for electronic mail systems.

Once a description has been obtained from a user, identification as an entity known to the system is also relatively simple; since each parameter to the service typically must be a member of a finite class known to the system. A directory assistance system, for example, would know the set of all listings that it could consult, and a restaurant reservations system, the date, time, and party size must be a member of a finite set. The name in which a reservation is made is an exception to this rule.

Even though the range of entities that can be described, and the method of identifying entities from their descriptions is much simpler than in general human conversation, the end results of attempting identification from descriptions can be classified into the same three categories: unique, ambiguous, and unsatisfiable. A gracefully interacting system must, therefore, be able to participate on either side of a dialogue for clarifying unsatisfiable or ambiguous descriptions; these dialogues will be the subject of the following subsections. Our examples and analysis in these subsections will be in terms of simple service domains, and no claims are made beyond this, Many of the strategies presented are, however, based on protocols of human conversations, and have much relevance to broader classes of human communication. We know of no other work that has attempted to analyse this type of

clarification dialogue.

### 7.2. Ambiguous Descriptions

When a speaker describes an entity to a listener in such a way that the listener can interpret the description as referring to more than one entity, the description is said to be ambiguous. In order for the speaker's attempt at description to succeed in such a situation, an clarifying dialogue must ensue to resolve the ambiguity. In this section we investigate the strategies involved in such a resolution, restricting ourselves to the case in which the descriptions are given to a gracefully interacting simple service system by its user.

The simplest strategy for ambiguous descriptions is to inform the user of the options and ask him to make a decision between them as in:

U: What is the number for Smith?
S: Do you mean Jim Smith or Joe Smith?

Note that in distinguishing the options by the first name of the person, the system refines the focus of the conversation to the corresponding slot in the description of a person.

Sometimes the alternatives may be too many to list and in this case the system can fix on a distinguishing feature of the alternatives and ask the user about it.

U: What is the number for Smith?
S: There are 59 Smiths listed. Do you know the first name?

While unavoidable in this case, this strategy has the disadvantage of inviting a wider variety of reply than the strategy of explicitly listing the alternatives. In general, some method for weighing the relative advantages of the various strategies of clarification according to the prevailing circumstances may be needed. Both strategies, however, change the focus to be a distinguishing aspect (slot) of the entities fitting the original ambiguous description. This focus shift serves as the basis for understanding a continuation of:

U: I think it starts with a "P"...

to the above example.

There is, of course, no guarantee that the user will choose to follow the system's shift of focus. He may instead try to resolve the ambiguity by giving distinguishing information of his own choosing; in particular, by further specifying an aspect of the entity different from the one chosen by the system.

U: I don't know the first name, but the address is Willow Crescent.

This corresponds to a shift of focus to a sibling node in the tree of possible foci. In any case, the system must be able to take whatever distinguishing information the user gives it

and apply it to the resolution of the ambiguity. If the ambiguity is still not resolved, yet more interaction must take place with the user.

Instead of trying to resolve the ambiguity, the user may also change the original description, giving

U: No!! I meant Smythe.

as a possible continuation for the last example, abandon the attempt at identification completely, or reiterate the ambiguous description. These cases must all be recognized separately, and appropriate responses given.

Occasionally, for domain dependent reasons, one of the alternatives for an ambiguous description may be much more likely than any of the others. In such cases, the alternative may be offered as a yes/no option to the user without referring to the others. Suppose reservations are available only after 9:30:

U: I'd like a reservation for two tonight at 8:30 or later.
S: Would 9:30 be OK?

Even though the user's specification covered 8:30 and all later times, 9:30 is the time of those available that the user is most likely to prefer. A similar strategy can be followed if the user is unlikely to be concerned about which of the alternatives is chosen.

There may also be user dependent reasons for the system to prefer one alternative referent over another. If the system has previous experience of a particular user, it might chose an alternative which it knows the user normally prefers. A restaurant reservation system, for instance, might have knowledge of the time a client usually prefers to dine. Again, knowledge of the current goals of a user might provide some clue to his choice between two alternatives. If a restaurant system knew, for instance, that the user wished to dine after the theatre, it might select an alternative reservation time that conformed to that constraint. We have not, however, considered the significant problems involved in using and acquiring idiosyncratic information, or in modelling and using an open-ended set of real-world goals.

### 7.3. Unsatisfiable Descriptions

In this section we consider descriptions from which a listener cannot identify any entity he knows about. The basic human strategy in such cases seems to be to indicate the unsatisfiability as well as indicating any near-misses, i.e. entities that "almost" fit the description. As before we will discuss unsatisfiable descriptions assuming that the listener is a gracefully interacting simple service system and the speaker is its user.

When a user's description is unsatisfiable by any entity consistent with the system's constraints, the basic strategy must be to inform the user of the problem and expect him to either change the description so that it is satisfiable or give up. In doing this, the system should provide the user with as complete a picture as possible of why the description is unsatisfiable. This information will help the user in reformulating his description, and, in particular, help him avoid giving another description that is faulty.

U: What is the number for Mr. Jim Smith?
S: I'm sorry! There is no listing for any Smith with initial 'J'.

U: I'd like a reservation for ten people tonight.
S: I'm sorry! we can't accommodate parties larger than seven..

U: I'd like a reservation for two for tonight.
S: I'm afraid the earliest reservation is next Thursday.

Implicit in these explanations of unsatisfiability is the search for "near-misses", entities which nearly fit the description. It is considered helpful for humans to suggest such near-misses, and a gracefully interacting system should do the same. Of course, any metric for measuring whether an entity is a near-miss must be quite domain dependent. The metric may also be dependent on the idiosyncracies of a particular user, or on user goals only distantly related to the domain of the system, in much the same way as the automatic selection between alternative possibilites for ambiguous descriptions discussed at the end of the last section. As also mentioned there, we have not attempted to address the formidable problems that these topics raise.

Near-misses may be either unique (one near-miss) or ambiguous (several comparably distant near-misses). They can be treated similarly to the basic unique and ambiguous description cases, except that it must be made plain that the description was satisfied by no entity as it stood. In addition, the unique case must be presented in a yes/no question to the user rather than assumed.

U: What is the number for Joe Smith?
S: We don't have a listing for Joe Smith. Do you mean Jim Smith?

U: What is the number for Joe Smith?
S: We don't have a listing for Joe Smith. There's one for
Jim Smith or Pete Smith or one for Joe Smythe.

In addition to the replies to the basic ambiguous case discussed above, the near-miss case can result in replies in which the user claims that the near-miss was what he said in the first place, or in which he repeats (possibly in the negative, "but there isn't any Joe Smith?") the original description.

### 7.4. Descriptions and Faulty Comprehension

In discussing the replies that a gracefully interacting system should make to ambiguous or unsatisfiable descriptions, we have assumed that the system comprehended the descriptions both fully and accurately. A partially or uncertainly comprehended description can, of course, be classified in the same way as unique, ambiguous, or unsatisfiable, but the way in which the system reacts to such descriptions should be modified in the presence of faulty comprehension. We can distinguish the two subcases of incomplete and uncertain comprehension.

If the system's comprehension of a description is incomplete, so that some aspects of the description are comprehended, while others are not, then the system's reaction ought or ought not to be modified depending on the classification of the description. If such a description is ambiguous and the missing part of the description might possibly resolve the ambiguity, then the system should assume that it would, and ask the user a suitable clarifying question.

U: What is the number for <GARBLE> Smith?
S: Did you say Jim Smith or Joe Smith?

If, on the other hand, the uncomprehended part of the description could not distinguish the two alternatives, then the description should be treated in the same way as an ordinary ambiguous description. For example, if there is a Jim Smith and a Joe Smith, both with the same title, then the following sequence is appropriate.

U: What is the number for <GARBLE> J. Smith?
S: Do you mean Professor Jim Smith or Professor Joe Smith?

In the same way, if we assume that uncomprehended parts of a description can only further restrict the class of entities to which a description could refer, there is no point in the system's trying to determine the uncomprehended part of a description whose comprehended part is already unique or unsatisfiable; for instance, there is no point in trying to determine the garbled part of "Professor <GARBLE> Smith" if only one Professor Smith is known to the system. Of course, there are exceptional cases (negatively phrased descriptions) in which this assumption does not hold, but normally it will allow the system to ignore the uncomprehended part without ill-effect (see Section 2.4).

Now we turn to the case in which a description is completely, but uncertainly, interpretable; the uncertainty can be about one interpretation or be between several possible interpretations. The action a system should take in such cases depends very much on the classification of the uncertain interpretations as either unique, ambiguous, or

unsatisfiable. If one of the alternatives is unique, while the others are all ambiguous or unsatisfiable, then this interpretation should be raised in certainty (on the grounds that people try to give unique descriptions), possibly to the extent that the system would want to accept this interpretation over all the others (perhaps checking it with an echo see Section 2.4). In general, being unique will tend to raise a description in certainty, while being ambiguous will tend to lower it, and being unsatisfiable will tend to lower it even more. The picture changes, however, if an unsatisfiable alternative has near-misses, especially if it has just one near-miss. In such a case it might be fruitful to re-analyse the input to see if the alternative could be reinterpreted as a direct description of the near-miss entity. If it could, then there would be reasonable grounds for assuming that it was the interpretation the user originally intended. The process of mapping descriptions onto entities must thus be closely coordinated with the process of analysing the descriptions linguistically.

### 7.5. Summary of Identification from Descriptions

We can summarize the main points of this section as follows:

- A fundamentally important important part of human conversational skills is the ability of a listener to identify entities from a speaker's description of them.
- In the given context, such descriptions may be uniquely specifying, ambiguous, or unsatisfiable by any entity known to the listener.
- In the ambiguous case, the speaker should be informed of the alternatives or asked for further distinguishing information, but there is no guarantee that he will choose to make the distinction in the way the listener suggests.
- In the unsatisfiable case, the listener should mention "near-misses", entities that almost fit the description.
- If the description was only partially or uncertanly comprehended by the listener, the strategies he uses for ambiguous and unsatisfiable descriptions should be modified in a number of detailed ways.

## 8. Language Generation

The extraordinary ability of humans to make sense out of the most convoluted and obscure language forms makes language generation one of the less critical components of graceful interaction; even quite clumsy utterances by a gracefully interacting system could be understood without great difficulty by its user. Nevertheless, a user cannot be presumed upon too much to compensate for a system's generative deficiencies; if a system's utterances are too clumsy, they will not appear natural; if they are over-detailed or too long winded, the user may become frustrated; in either case, the interaction will be less graceful than it could

be.

In the remainder of this section, we discuss briefly some of the characteristics of human language that a gracefully interacting system should imitate in order to appear natural, including contextually dependent (anaphoric) references and ellipsis, and the inclusion of several different messages (illocutionary acts) in the same utterance, especially combining echoes with other output. We will also discuss generation considerations specific to gracefully interacting systems; in particular, the need to restrict the complexity of generated utterances in accordance with analytical ability.

When a human describes some entity, he does not normally incorporate all that he knows about the entity into the description; rather, as Grice [12] has pointed out, he normally uses sufficient information to identify the entity to his listener, but no more. Thus, if a speaker wanted to describe a black desk, he might just say "the desk" if he thought that description was sufficient for his listener to understand what he meant. On the other hand, if he thought there was some danger of confusion with a different desk (a brown one, say), he might say "the black desk". The way an entity is described in human conversation, then, depends not only on the entity and the speaker's knowledge of it, but also on the speaker's beliefs about his listener's state of knowledge and focus of attention. An appropriate description is typically the least informative one that is sufficient to identify the described entity relative to the listener's current focus of attention. Little attention has been paid in natural language research to generating descriptions appropriate in this sense, except for the description generation system of Levin and Goldman [26].

The problem is, nevertheless, important for a gracefully interacting system; if the descriptions such a system uses to identify objects to its user do not conform to human conventions, the user will find them unnatural and ungraceful. Thus, if a restaurant reservations system wishes to offer its user a reservation on Wednesday at 7pm, and if the user has already made it clear that he wants a reservation on Wednesday in the evening, the system need only and should only describe the reservation by "7" (as in "Would 7 be ok?").

Using these conventions leads to the generation of anaphoric references. A similar convention (of not giving more information than one has to) can result in elliptical utterances, e.g. replying "Jim Smith" instead of "I mean Jim Smith" in response to "Do you mean Jim Smith or Fred Smith?". It is clearly desirable for a gracefully interacting system to use its knowledge of the goal structure and focus of attention of a conversation to generate such ellipses and anaphora as well as analyse them (see Section 6.3).

Humans often use a single utterance to serve several different purposes (or following Searle's terminology [36] to perform several illocutionary acts). A person might, for instance,

ask if a window was open, both to indicate that he was cold and to ask his listener to shut any open window, as well as to find out whether a window was indeed open.

At a much less sophisticated level, an ability to combine more than one illocutionary act in a single utterance is also useful for a gracefully interacting system. In particular, such an ability is useful for the kind of indirect echoing described in Section 2.4. In indirect echoing, a listener who is unsure of his interpretation of a speaker's utterance incorporates his uncertain interpretation into his reply. If the original speaker does not comment on the echo, the interpretation is implicitly confirmed. Thus, if a restaurant reservations system was unsure about its recognition of "seven" in:

I'd like a reservation for seven people.

a suitable reply would be:

What time would your party of seven like to eat?

The substitution of "your party of seven" for the more normal "you" represents an indirect echo of the system's uncertain interpretation. Of course, using indirect echoes can (and in this example did) violate the convention discussed above of using descriptions only minimally sufficient for identification. However, this appears to be acceptable in human dialogue, presumably because of the clarification purposes thus served. Similarly, indirect echoes can also replace ellipses with fuller forms, as in:

U: I want to go to Pittsburgh on May 17.
S: What time on May 17 do you want to go to Pittsburgh?

instead of:

S: What time do you want to go?

in which the date and destination are ellipsed.

Before ending this section, we must consider an aspect of generation that is much more important for gracefully interacting systems than in general human conversation. As mentioned in Section 2.3, the form of a speaker's utterance tends to influence the form of the listener's response. Thus it is vitally important for a gracefully interacting system with a limited ability to comprehend language to co-ordinate the questions and other output it generates with its receptive repertoire. This co-ordination has two parts: first, a system should try to restrict the content of the user's reply by asking questions that are as specific and directive as possible. To repeat an example from Section 2.3, if a system cannot understand "extension" in "I would like the extension for Jim Smith", it should ask "Do you want the number for Jim Smith?", rather than "What do you mean by 'extension'?". Secondly, a system should be able to understand standard transformations and convolutions of the phrasings in which it asks its questions. Thus, if a system asks "Would you prefer 7 o'clock

or 8 o'clock?", it should certainly be able to understand "I prefer 7 o'clock.", and should probably also understand convolutions as distant as "7 o'clock would be my preference.".

In summary, although a human user can be expected to compensate somewhat for the deficiencies of the output produced by a gracefully interacting system, the more natural the output is, the more graceful the system will appear to the user. In particular, the system should use its model of the focus to generate elliptical and anaphoric utterances when possible. However, sometimes other demands, particularly those of indirect echoing, can override this requirement for terseness and cause the information omitted from an utterance by ellipsis and anaphora to be reintroduced. Finally, a gracefully interacting system should produce only output that it can understand itself, in case it is transformed by the user and incorporated into his reply.

## 9. Realizing Graceful Interaction

Having outlined the basic components of graceful interaction, we now turn to the question of how to fit the components together in a single gracefully interacting system. In the following subsection, we give some design considerations for complete gracefully interacting systems, and go on to outline an architecture in line with these considerations for graceful interaction in simple service domains. We are planning to implement a practical gracefully interacting interface according to the proposed architecture, but at the time of writing have only just started to do this. The architecture proposed must therefore be regarded as tentative and subject to revision. A second subsection follows a hypothetical gracefully interacting system employing the proposed architecture through a realistic example dialogue in which many of the components of graceful interaction that we have discussed come into play.

### 9.1. Architecture of a Gracefully Interacting System

The overall design of any gracefully interacting system must take into account three important characteristics of the interplay between the various components of graceful interaction.

- At any point in the type of conversation we are considering, the role of the gracefully interacting system can be described in terms of just one of the components of graceful interaction.

- The several components of graceful interaction come into play in an inherently asynchronous manner, in the sense that a system's use of one component may be interrupted by use of one of the others without the first coming to a proper conclusion.

- Components interrupted in this way can often be restarted from where they left off after the interruption is over, but this is not inevitable.

Consider, for example, a dialogue concerned with identification of some entity by the system from a user's description. While this dialogue is taking place, the behaviour of the system can be explained purely in terms of the identification from description component. Yet at any time this line of conversation can be interrupted in favour of (say) a channel-checking dialogue, or an echo correction, or even an identification from a different description (see Section 6.4). Moreover, the original identification may or may not be taken up after the interruption is over.

Because the different components of graceful interaction can come into play in this inherently asynchronous manner, we propose to organize our gracefully interacting system as a set of autonomous modules, each capable of carrying on a particular type of dialogue: one for channel checking, one for identification from descriptions, etc.. At any time, exactly one of these modules would be active, in the sense that its state would hold the system's primary view of what was happening in the conversation. However, the other modules would be kept in a state of readiness, and if the user's next input could be dealt with better by one of the other modules, it would be made the currently active module. The state of a module usurped in this way would, of course, be preserved in anticipation of a possible continuation. Each module would indicate which inputs it was able to deal with by a set of expectations or conditions; the inputs it can deal with are those that satisfy the expectations. These expectations could change according to what had gone before; thus, a description identifier module would have different expectations after it had presented a list of alternative fillers of an ambiguous description, than after it had informed the user that a description was unsatisfiable; again, a module for detecting echo corrections would only have an expectation at all after the system had issued an echo.

What modules are needed, and what are their functions? We give here our current answer for simple service systems; we expect that most of the modules would stay the same for a gracefully interacting system of greater abilities.

- **Channel maintainer:** This module can conduct a channel-checking dialogue. It will initiate one if the user fails to reply to the system, and will respond to the user's initiation of such a dialogue.

- **User's echo monitor:** This module can conduct a dialogue to correct incorrect echoes by the user of what the system said. It will detect any echoing by the user and initiate a correction dialogue if the echo is incorrect.

- **Echo correction monitor:** This module can detect any correction by the user of echoes by the system. It will take part in a dialogue to ensure that the

correction is understood properly.

- **Explicit incomprehension monitor:** This module can engage in a dialogue to clear up explicit complaints of complete or partial misunderstanding by the user. It will initiate its dialogue when the user makes such an explicit complaint.

- **Incomprehension resolver:** This module is used whenever the degree of comprehension of the user's input is unsatisfactory. According to the degree of incomprehension and the importance of what was not understood, it can echo or request clarification explicitly, and participate in any clarifying dialogue. (The echo correction monitor could be included as part of this module, since it deals with a logical continuation of echoing by the system).

- **Description identifier:** This module is used to identify entities from descriptions by the user. If necessary, it can request descriptions explicitly, and engage in dialogue to clarify ambiguous or unsatisfiable descriptions.

- **Ability and goal describer:** This module can engage in a dialogue about the system's ability, either in response to direct questions, or as a last resort when incomprehension resolution does not appear to be working. It can also describe to the user what it thinks the user is trying to do, how it is cooperating with that, and what the user needs to do to let it succeed; this information is available through the focus of the system.

- **Initiator:** This module is used to establish communication between the system and the user. It is always the first module activated. Besides ensuring that the communication channel is indeed open, it can conduct a dialogue about what the system is and can do; this allows the user to make certain that the system is the party he really wants.

- **Terminator:** This module is used when the conversation needs to be terminated. It can initiate a termination when the task has been completed, and can also be activated when the user decides to give up before finishing the task.

To coordinate all these modules, some kind of executive is needed. The job of this executive is to direct the user's input to the appropriate module, and to make sure that when one module completes its dialogue, control is turned over to the next appropriate module. For these reasons we call this executive module the focus maintainer. The focus maintainer we envisage is tailored directly to simple service systems, and assumes a frame-based representation as discussed in Section 4.3. It can handle input containing descriptions of more than one slot by parcelling the descriptions out to appropriate invocations of the description identifier module; it keeps track of which slot the user and system are currently trying to fill; it can detect any correction by the user of slots that have previously been filled; it can initiate filling of a slot when the user has not tried to fill it; it can deal with descriptions of several slots at once; it can decide when all slots have been filled; and it can keep track of the focus within a slot, i.e. which aspect of the entity described is being used to resolve ambiguities.

Parsing of each input will be done in accordance with the demands of flexible parsing discussed in Section 3.2; in particular, the parsing will be bottom-up, not strictly directional, and based on pattern-matching. Each module will parse its input separately (if this results in too much duplication of effort, partial results could be saved). This allows the expectations of each individual module to influence the parsing in two advantageous ways: first, at any given time the parser will only use patterns relevant to the module for which it is parsing; and secondly, expectations of specific replies can be used to recognize ellipses as complete utterances in themselves, as discussed in Sections 3.2 and 6.3. Using these strategies, input from the user that satisfies an active module's expectations can be recognized with a minimum of search.

How should the several modules listed above be implemented? One possibility is to model each of them as a finite state network. As was shown in the discourse component of the Hearsay system [19], a finite state network is very suitable for representing a straightforward conversation in which nothing unexpected happens. The preceding dialogue can be represented implicitly by the state of a finite network plus possibly the contents of some registers, and transition to other states can be made conditional on both the input and the contents of registers, thus ensuring that the state reached by a transition reflects all the relevant history of the conversation, and enabling the system's response to a user's input to depend both on the input, and on what has gone before in the dialogue.

These features appear to make the finite state model extremely suitable for the several modules described above, since each of them can, by definition, conduct a straightforward conversation in which nothing unexpected happens. Figure 1 shows a finite state net for a simplified description identifier module; the diamond boxes are tests and the square boxes are actions, while the circles are states; transitions are made from state to state according to the conditions, which typically depend on the input, but in some cases also depend on registers (e.g. the test for one option specified, after the ambiguous description state). It is easy to see from the example that the expectations of a given module in a given state correspond to the conditions on the arcs leading from that state.

Nevertheless, the finite state model for the individual modules suffers from certain disadvantages. These arise mainly because the implicit way in which state information is represented in a finite state network. In such a network it is natural to encode current goals and past history implicitly in the state nodes. In Figure 1, for example, the state "ambiguous description" implicitly encodes the fact that the user has given the system an ambiguous description, and the goal of the system to get the user to make that description unambiguous. This implicitness makes finite state networks relatively hard to modify, since the state implicit

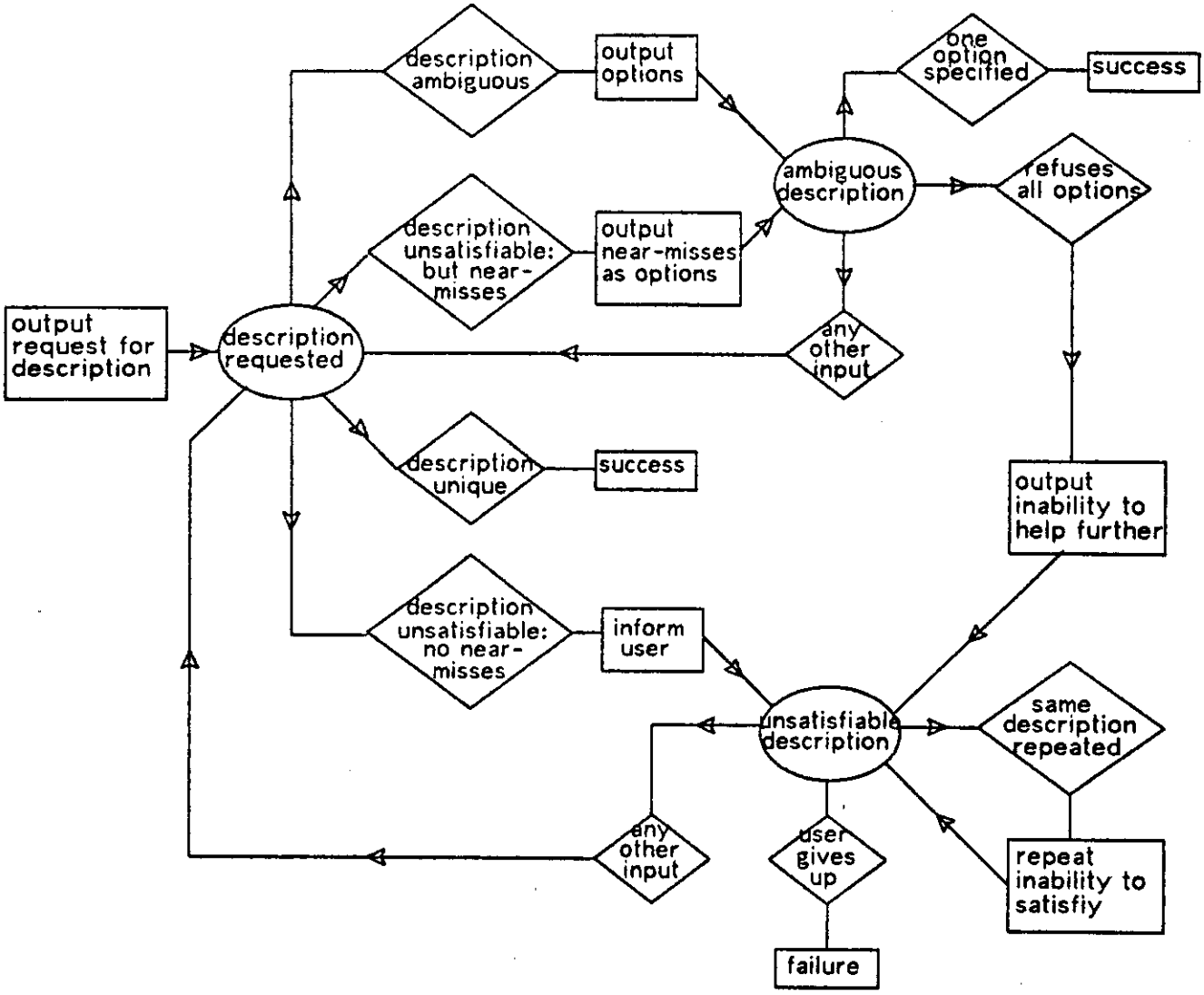


Figure 1: Finite-state net for description identifier module

in a node must in general be taken into consideration when links to or from that node are added or deleted. More serious perhaps is the problem that finite state modules would not be perspicuous to other modules, and in particular to the focus maintainer. We believe that communication between modules about the goals that they are currently pursuing may be very important to a complete system. It may be best to tackle the problem of modifiability by representing each module as an independent set of production rules. It should still be possible to compile such rule sets into finite state networks if extra efficiency proved important. Communication between modules might be achieved in the context of independent rule sets through shared global variables. On the other hand, it may be necessary to maintain

a stack of system and user goals in a way similar to the "dialogue games" model of Levin and Moore [25]. We expect to resolve these questions and many other more detailed ones through experience with the implementation that we are currently undertaking along the lines presented above of a gracefully interacting system for composing, sending, filing, and retreiving electronic mail.

### 9.2. Worked Example

Below we list a realistic example of the sort of dialogue that we expect a gracefully interacting system to be able to participate in; The example was constructed to illustrate most of the components of interaction that we have discussed. The system in the example is a simple service system for restaurant reservations. We will follow through the dialogue and see how the several modules described above contribute to the system's participation.

S: Hello! This is the Brown Cow restaurant.
U: Hello! The Brown Cow restaurant?
S: Yes! Would you like to make a reservation?
U: Yes! I'd like one for eight tonight.
S: At eight o'clock, and for how many people?
U: There will be (five) of us.
<clicks on line>
U: Hello! Hello! Are you there?
S: Yes! Can you hear me?
U: Yes I can.
S: OK. For three people, and what name will that be under?
U: No! I said five.
S: Oh! Five people. I can't give you eight o'clock; it would have to be seven or after nine thirty.
U: You don't have anything between those times?
S: No! I'm afraid we don't.
U: Well, we'll wait until the later time. Was it nine or nine thirty?
S: It was nine thirty. What name will that be under?
U: Teacon.
S: Was that Beacon with a 'B', or Deacon with a 'D'?
U: It's Teacon with a 'T'; T E A C O N.
S: T E A C O N. Very good! We'll expect five of you at nine thirty.
U: OK. Thank you!
S: You're welcome! Bye!
U: Bye!

The first three lines are dealt with by the initiator module. A conversation is always started by the activation of this module, and its first action is to give a greeting which both informs the user that the channel of communication is open, and tells him who he is talking to. On line 2, the user asks for some reassurance that he is talking to the right party; the module provides it, de-activates itself, and passes control back to the focus maintainer; Since the focus maintainer has no input to work on, it asks the user if he wants to use the (only)

service that the system can provide.

If, at this point, the user had still not been satisfied with whom it was taking to, the initiator module would have been re-activated. Instead, the user satisfies the expectation of the focus maintainer by specifying a reservation. We are assuming a frame-based system, so the focus maintainer must now activate the description identifier module for each slot covered by the specification. Since the two slot specifications are uniquely identifying, the identifier module terminates in both cases without generating any output. However, the prior choice of which slots to use was not straightforward: "tonight" specifies the date slot, but "for eight" could specify either the party size or the time. The focus maintainer has two options: either ask the user explicitly which one he meant (using the incomprehension resolver module), or choose one of the options on its own initiative. Since eight is a sufficiently large party size, and a sufficiently popular reservation time to make the time interpretation significantly more plausible, it makes its own choice. However, an assumption like this cannot be made without informing the user, so the assumption is included in an echo at the start of line five. Making the echo will set up an expectation by the echo correction monitor that the user will correct the assumption; however, in this case the system struck lucky; the echo is not challenged, and the assumption is thereby implicitly confirmed.

After echoing, the focus maintainer selects an empty slot, the party size, and starts the description identifier for that slot, which in turn asks for the party size. The user's reply conforms to the expectations of the now active description identifier by specifying unambiguously a party size of what the system hears as three, but which really is five. However, the system is sufficiently unsure of its perception of the number to mark it down for an echo.

At this point, an unexpected sound on the line causes the user to initiate a channel checking dialogue. This input satisfies the ever-present expectations of the channel maintainer, so that it becomes active at this point and conducts the dialogue up until the OK of line 11. The conflict between the satisfaction of the expectations of both the channel maintainer, and the description identifier is resolved by a precedence ordering on the several modules; the precedence is analogous to an interrupt priority scheme in which the most pressing concerns receive the highest precedence, so that the channel maintainer has the highest precedence, followed by the echo modules, with the description identifier having the lowest precedence.

Now that the channel has been re-established, the description identifier can continue from where it left off, so it echoes the party-size, and then de-activates itself, allowing the focus maintainer to take over. The focus maintainer then selects the only slot still to be filled, and

starts up the description identifier on that slot, resulting in the request for the name. However, the user has detected the incorrect echo and issued a correction. This does not satisfy the expectations of the active description identifier, but does satisfy those of the echo correction monitor. The monitor corrects the description, and, after re-echoing to ensure that the message has been received correctly this time, terminates, allowing the focus maintainer to restart the previous invocation of the description identifier with the new description.

Unfortunately, the description is no longer satisfiable, since no table for five people is available at eight o'clock. In this situation, the normal response of the description identifier would be to look for near-misses for the party-size, but it would clearly be a mistake to offer the user a table for three or four at eight o'clock. The problem arises because of the interaction between the party-size and the time in determining whether the description is satisfiable. The system, therefore, provides a method of specifying which slots of a frame constrain which others, and offers a mechanism for specifying which to change to resolve any conflicts that arise. In this case, the time, date, and party-size all interact to constrain each other, and changes are to be attempted in that order. The net effect is that instead of indicating that five is an unsatisfiable description of a party-size, the description identifier for party-size accepts five as a unique description, terminates, and restarts the description identifier for time with the previously given description.

This time the normal procedure for dealing with unsatisfiable descriptions applies, and the system lists its near misses. The user's reply in line 14 satisfies one of the expectations of the description identifier by essentially repeating the same specifications. The reply is a straightforward affirmation that the limitations given are truly correct. The tendency for humans to ask for confirmation when they hear something that they don't want to hear is very common, and there should perhaps be a larger recognition of that in a gracefully interacting system, but we have not investigated that possibility.

The user next accepts the later of the two times, but asks the system to confirm exactly which time it is. The first part of this input terminates the description identifier for the time, and the second part is handled by the explicit incomprehension monitor, which generates the first part of line 17. The second part is generated by a re-initialized version of the description identifier for the name in which the reservation is to be made. The module is re-initialized because a module cannot be continued from where it left off if other modules at the same priority level have participated in the intervening dialogue. The ensuing determination of the name shows some of the special problems that can arise when proper names not previously known to the system have to be communicated. To understand such descriptions properly, a system needs a catalogue of common names and an ability to recognize spellings.

Finally on line 21, the reservation is complete and the system reconfirms the essential points. Again this is an echo which the user could contradict. In the event, he is satisfied and the conversation is finished off by the terminator module.

## Conclusion

Graceful interaction is of fundamental importance for all computer computer systems that interact with humans. Without graceful interaction skills, interactive computer systems will continue to appear uncooperative, uncompromising, and altogether obtuse to the non-expert user. Yet, as we have seen, graceful interaction is a little studied field; some work has been done on some of the skills required, but such efforts have generally been tangential to the main thrust of the systems in which they were embedded. No workers have attempted to study the graceful interaction problem as a whole.

In this paper we have attempted to circumscribe graceful interaction as a field for study. To this end, we have defined a set of basic components of graceful interaction, including robust communication, flexible parsing, explanation facilities, focus mechanisms, identification from descriptions, and generation of descriptions. These components are certainly necessary for any kind of graceful interaction, but are also sufficient, we claimed, for systems restricted to offering one of the very large class of simple services.

Finally, we believe that graceful interaction is an idea whose time has come, and is ripe for implementation. We claim that none of the components we described is significantly beyond the current state of the art in dialogue processing (at least not for simple service systems), and we have presented an architecture for their integrated implementation in a single gracefully interacting system. A gracefully interacting simple service system (for sending and receiving electronic mail) conforming to this architecture is currently under implementation.

**Acknowledgements**

We are particularly grateful to Jaime Carbonell and Candy Sidner for their insightful comments on earlier versions of this paper.

## References

1. Austin, J. L. *How to Do Things with Words.* Oxford University Press, 1962.

2. Bobrow, D. G., Kaplan, R. M., Kay, M., Norman D. A., Thompson, H., and Winograd, T. GUS: a Frame-Driven Dialogue System. *Artificial Intelligence 8* (1977), 155-173.

3. Carbonell, J. G. *Subjective Understanding: Computer Models of Belief Systems.* Ph.D. Th., Yale University, 1979.

4. Carbonell, J. R. Mixed-Initiative Man-Computer Dialogues. tech report 1970, Bolt, Beranek, and Newman, Inc., 1971.

5. Charniak, E. C. Toward a Model of Children's Story Comprehension. TR-266, MIT AI Lab, Cambridge, Mass., 1972.

6. Codd, E. F. Seven Steps to RENDEZVOUS with the Casual User. In Klimbie, J. W. and Koffeman, K. L., Ed., *Proc. IFIP TC-2 Working Conf. on Data Base Management Systems*, North Holland, Amsterdam, 1974, pp. 179-200.

7. Cohen, P. R. *On Knowing What to Say: Planning Speech Acts.* Ph.D. Th., Dept. of Computer Science, University of Toronto, January 1978. also available as Tech. Report 118

8. Colby, K. M. Simulations of Belief Systems. In Schank, R. C. and Colby, K. M., Ed., *Computer Models of Thought and Language*, Freeman, San Francisco, 1973, pp. 251-286.

9. Doyle, J. Truth Maintenance Systems for Problem Solving. TR-419, MIT AI Lab., Cambridge, Mass., 1978.

10. Fahlman, S. E. *NETL: A System for Representing and Using Real-World Knowledge.* MIT Press, Cambridge, Mass., 1979.

11. Fox, M. S., and Mostow, D. J. Maximal Consistent Interpretations of Errorful Data in Hierarchically Modelled Domains. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 165-171.

12. Grice, H. P. Logic and Conversation. In Cole, P. and Morgan, J. L., Ed., *Syntax and Semantics: Speech Acts*, Academic Press, New York, 1975.

13. Grimes, J. E. Topic Levels. Proc. of Theoretical Issues in Natural Language Processing: An Interdisciplinary Workshop, University of Illinois at Urbana-Champaign, July, 1978, pp. 104-108.

14. Grosz, B. J. The Representation and Use of Focus in a System for Understanding Dialogues. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 67-76.

15. Grosz, B. J. Focusing in Dialogue. Proc. of Theoretical Issues in Natural Language Processing: An Interdisciplinary Workshop, University of Illinois at Urbana-Champaign, July, 1978.

16. Hayes, P. J. On Semantic Nets, Frames, and Associations. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 99-107.

17. Hayes, P. J. A Representation for Robot Plans. Proc. Fourth Int. Jt. Conf. on Artificial Intelligence, Tbilisi, Georgia, 1975, pp. 181-188.

18. Hayes-Roth, F., Erman, L. D., Fox, M., and Mostow, D. J. Syntactic Processing in HEARSAY-II. Speech Understanding Systems. Summary of Results of the Five-Year Research Effort at Carnegie-Mellon University, Carnegie-Mellon University Computer Science Department, 1976.

19. Hayes-Roth, F., Gill, G., and Mostow, D. J. Discourse and Task Performance in HEARSAY-II. Speech Understanding Systems. Summary of Results of the Five-Year Research Effort at Carnegie-Mellon University, Carnegie-Mellon University Computer Science Department, 1976.

20. Hendrix, G. G. Human Engineering for Applied Natural Language Processing. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 183-191.

21. Hendrix, G. G. Expanding the Utility of Semantic Networks through Partitioning. Proc. Fourth Int. Jt. Conf. on Artificial Intelligence, Tbilisi, Georgia, 1975, pp. 115-121.

22. Hockett, C. F. *A Course in Modern Linguistics.* MacMillan, New York, 1959.

23. Kaplan, S. J. *Cooperative Responses from a Portable Natural Language Data Base Query System.* Ph.D. Th., Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, 1979.

24. Lehnert, W. *The Process of Question Answering.* Lawrence Erlbaum, Hillside, N. J., 1978.

25. Levin, J. A., and Moore, J. A. Dialogue Games: Meta-Communication Structures for Natural Language Understanding. *Cognitive Science 1,* 4 (1977), 395-420.

26. Levin, J. A. and Goldman N. M. Process Models of Reference in Context. ISI/RR-78-72, Information Sciences Institute, Univ. of Southern California, October, 1978.

27. Mann, W. C., Moore, J. A., and Levin, J. A. A Comprehension Model for Human Dialogue. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 77-87.

28. McKeown, K. Paraphrasing Using Given and New Information in a Question-Answering System. Association for Computational Linguistics Meeting, San Diego, 1979.

29. Minsky, M. A Framework for Representing Knowledge. In Winston, P., Ed., *The Psychology of Computer Vision,* McGraw Hill, 1975, pp. 211-277.

30. Parkison, R. C., Colby, K. M., and Faught, W. S. Conversational Language Comprehension Using Integrated Pattern-Matching and Parsing. *Artificial Intelligence* 9 (1977), 111-134.

31. Raphael, B. SIR: A Computer Program for Semantic Information Retrieval. In Minsky, M. L., Ed., *Semantic Information Processing,* MIT Press, Cambridge, Mass., 1968, pp. 33-134.

32. Riesbeck, C. K. Computational Understanding: Analysis of Sentences and Context. Working Paper 4, Istituto per gli Studi Semantici e Cognitivi, Castagnola, Switzerland, 1974.

33. Sacerdoti, E. D. Language Access to Distributed Data with Error Recovery. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 196-202.

34. Sacerdoti, E. D. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence 5,* 2 (1974), 115-135.

35. Scragg, G. W. *Answering Questions about Processes.* Ph.D. Th., U. of California, San Diego, 1974.

36. Searle, J. R. *Speech Acts.* Cambridge University Press, 1969.

37. Sidner, C. A Progress Report on the Discourse and Reference Components of PAL. A. I. Memo. 468, MIT A. I. Lab., 1978.

38. Sussman, G. J. and McDermott, D. V. From PLANNER to CONNIVER - a Genetic Approach. Proc. Fall Joint Computer Conf., AFIPS Press, Montvale, N.J., 1972, pp. 1171-1179.

39. Walker, D. Speech Understanding Research, Final Report, Project 4762. Artificial Intelligence Center, Stanford Research Institute, Menlo Park, Ca., 1976.

40. Waltz, D. L. An English Language Question Answering System for a Large Relational Data Base. *Comm. ACM 21*, 7 (1978), 526-539.

41. Weischedel, R. M. and Black, J. Responding to Potentially Unparseable Sentences. tech. report 79/3, Dept. of Computer and Information Sciences, University of Delaware, 1979.

42. Weizenbaum, J. ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine. *Comm. ACM 9*, 1 (January 1966), 36-45.

43. Wilks, Y. A. Preference Semantics. In Keenan, Ed., *Formal Semantics of Natural* Language, Cambridge University Press, 1975.

44. Winograd, T. *Understanding Natural Language.* Academic Press, New York, 1972.

45. Woods, W. A. Transition Network Grammars for Natural Language Analysis. *Comm. ACM* 13, 10 (October 1970), 591-606.

46. Woods, W. A., Kaplan, R. M., and Nash-Webber, B. The Lunar Sciences Language System: Final Report. 2378, Bolt, Beranek, and Newman, Inc., 1972.

47. Woods, W. A., Bates, M., Brown, G., Bruce, B., Cook, C., Klovstad, J., Makhoul, J., Nash-Webber, B., Schwartz, R., Wolf, J., and Zue, V. Speech Understanding Systems - Final Technical Report. 3438, Bolt, Beranek, and Newman, Inc., 1976.