# Discourse Strategies for Conversation in Time

James Raymond Davis
Media Laboratory
Massachusetts Institute of Technology
E15-325
Cambridge, MA 02139

617-253-0314
jrd@media-lab.media.mit.edu

and

Chris Schmandt
Media Laboratory
Massachusetts Institute of Technology
E15-327
Cambridge, MA 02139

617-253-5156
geek@media-lab.media.mit.edu

### Introduction

Speech takes time and takes place over time. As a consequence, conversational computer systems must be very different from graphical user interfaces, which are designed around the assumption that the graphics operations are more or less instantaneous, in response to user input happening sporadically over time. Conversational programs must account for the duration of an utterance and the history of the conversation. In deciding what to say now, a program should consider what it has said in the past and what it is likely to say in the future. This assumes, of course, that the program is sufficiently powerful that it chooses for itself what to say and how to say it. This paper considers these issues in general, and in particular as they affect the Back Seat Driver, a program which gives real time driving instructions.

## Deciding what to say

First and foremost, a program should say as little as possible[1]. All things being equal, we would rather a machine be silent - and this will be true until we produce machines with either more pleasant voices or more brilliant thoughts than anything in the laboratory today. Systems which engage in only the simplest kinds of dialogs can simply be designed for maximum brevity, but those with more powerful uses of language will require the ability to decide at run time what to say and how much to say about it. This will especially be true for programs which pursue more than one goal at a time. Such programs will often have more than one possible thing to say. In this case, they may choose to be brief because time spent on one goal is necessarily taken from some other.

### Choosing what to say

If a program can not meet all goals with a single utterance, it must then have some means of choosing which goal to pursue. For an interactive program this is even more crucial, because the user can add a new goal or cancel (or just make impossible) an existing one at any time. Choosing what to work on is an instance of the more general problem of efficient resource allocation, made more acute by the fact that time can not truly be "saved" - any moment that a program does not speak is lost forever.

Having chosen a goal, a program will sometimes find it must change that choice while working on it, that is, while speaking, and for speech, this means interrupting itself. This is undesirable because it wastes time and takes extra work. It wastes time because a partially spoken utterance can not reliably convey anything[2] so the time devoted to the interrupted utterance is wasted - and since the user gained nothing for her time listening, she is probably annoyed. It takes extra work because the interruption needs to be marked in some way - either by a cue phrase (e.g. "Ooops" or "That reminds me") as in [8] or by a change in intonation[3], as in [6]. To return to the interrupted subject later may also require a cue phrase (e.g. "anyway", or "as I was saying").

[1] Many current telephone "information services" violate this rule by adopting overly polite phrasing, even those which are free. Since "pay" phone services (i.e. 900 or 976 numbers) might increase revenue by increasing the duration of the call, perhaps they should reverse the strategies we describe below.

[2] This is for lack of a good linguistic or psychological theory. We know that people can communicate even when interrupting themselves and each other but we do not know how to make a program do this.

[3] Current speech synthesizers are not capable of smoothly stopping and restarting speech, so almost any effort to cause an interruption will result in some kind of audible boundary in the speech, if not a natural one. This is one advantage to the current poor state of machine prosody.

Since abandoning an utterance is undesirable, it follows that a planning policy should try to avoid pursuing goals that are likely to be discarded. Depending on the goals and task it may be better to speak only when reasonably confident that the utterance can be completed. This requires that the goal control mechanism have some way of estimating the probability of interruption.

Conversational systems which can interrupt themselves require a speech synthesis system able to stop speaking asynchronously without making unnatural sounds. It is also desirable that the synthesizer be able to report at any time whether it is presently speaking and, if so the position in the input of the last word spoken. It is also desirable that the speech rate be sufficiently predictable that the program can estimate the time required for speaking an utterance without having to actually speak it. Note also that current (analog) telephony makes it difficult for a program employing speech recognition over the telephone to understand speech that occurs while the program is talking, because the programs own audio output is fed back (albeit at a reduced level) to the recognizer through the telephone. This may improve when the telephone network is all digital.

## How to be brief

A program can reduce the time required for an utterance by speaking more quickly, provided this does not reduce intelligibility. There are other means as well. A program can also be brief by saying less; either conveying less information or using fewer words to do so. Saying less means omitting detail. For instance, a program might answer the question "Where are my keys?" saying either "Your keys are on your desk, next to the terminal, behind your book" or just "on your desk". This example shows both a reduction in detail and a change from a full sentence to a sentence fragment.

Note that the reduced form is not only less informative (though perhaps still sufficient) it is also has less redundancy. The advantage of the full sentence form is that it not only answers the question, it also shows how the answerer understood the question. Given that current speech recognition systems are not fully reliable, it may be desirable to use full sentence answers, when time allows, just to make systems more robust, as in [9].

Moreover, the reduced forms may be less intelligible to the human either because of this same lack of redundancy or because too much has been omitted. This requires that programs be prepared to repeat themselves on demand. When repeating, programs should speak more slowly, provided this does not introduce unnatural prosody in the synthesizer. Another possibility is to generate an alternative phrasing for the message to be repeated. Such a paraphrase should always be marked as such, lest the user the confused.

Note that humans are capable of great economy in their speech, partially because of their ability to make powerful inferences, and partially because they have powerful means of checking that references are correctly understood, identifying trouble spots, and repairing them, as for example described in [2]. It will be a long time until programs have such abilities, however.

## Discourse is the history of a conversation

Programs must also consider the history of the conversation - the discourse - in deciding what to say. Limited space here does not allow us to describe any of the many theories of discourse developed by linguists. We refer the interested reader to [1, 5]. A proper model of discourse structure is necessary if a program is to generate (or comprehend) texts greater than a single sentence. Without a discourse model, each sentence a program generates must stand on its own, and each sentence of input must be similarly treated in isolation.

The term “history” should not be understood to imply that discourse is simply a sequential list of utterances. Discourse structures include the purposes of utterances and the relations among those purposes, and tracks the concepts and objects of the domain under discussion as they come into focus.

Actually, discourse as it is currently understood is too narrow a term. What is actually needed is a structure that includes all kinds of actions, not simply acts of speaking. Current theories of discourse structure treat linguistic actions only, and not actions in general. Thus there is no way to account for utterances like “Stop doing that” where *that* refers to some ongoing action.

### Referring to time

The program may also need to refer to time itself. A familiar example is the telephone “time service”. Here the speech is referring to a specific moment - the “now”. If the hearer must take an action at the specified time, then some kind of warning or countdown is required. One reason, we believe, for the incoherence of messages on telephone answering machines is that the “beep” catches people by surprise. Even though they are warned to expect it, there is no clue about exactly when it will come. By contrast, people seem to be able to give good cues for timing by making rather complicated adjustments to pitch and speech rate while they talk. Such adjustments are beyond the present state of the art for speech synthesis.

Programs undertaking lengthy operations should inform their users of the expected duration, if longer than a few seconds, and should inform the user of the program's progress while working [7]. This is especially true for speech systems, which present no other indication of progress while working. A program which is capable of pursuing a second goal “in the foreground” may do well to make this explicit, perhaps by means of a cue phrase, as in:

> I've started the first file transfer going.
> It will probably take a few minutes.
> *Meanwhile*, do you want to read your new mail?

## The Back Seat Driver

The Back Seat Driver is a program which gives real time driving instructions to the operator of a car. Only a brief description is possible here, but see [3]. The Back Seat Driver is installed in a car. A navigation system provides it the current position twice a second. It finds routes using a digital street map, and speaks using a Digital Equipment Corporation DECTALK. As a program which pursues multiple goals over an extended time period in a moving vehicle, the Back Seat Driver illustrates all the issues described above.

The Back Seat Driver has many goals. Two of greatest importance are to keep the driver safe by giving warnings of upcoming traffic hazards and to get the driver to his or her destination. Lesser goals include informing the driver of her present location and providing background information about the current locale.

The Back Seat Driver is able to make an efficient choice of goal because each goal has four state variables which are helpful for scheduling. The **priority** is an integer which tells the importance of the goal. It is assigned by the programmer and does not change. The **progressable** flag is a Boolean variable which tells whether the goal can accomplish anything at the present time. For instance, the goal which warns of upcoming traffic hazards only needs to run if some hazard exists ahead and has not already been mentioned. Each progressable goal also estimates the **maximum speech time**, which is the number of seconds the program would require for delivering its message, were it to be allowed to speak at this time. Goals calculate this by formulating a message, counting the words in it, and dividing by the synthesizer speech rate. Finally, each goal which is not currently progressable estimates its **minimum silence time**, which is the number of seconds the goal is confident it will have nothing to say. Given this information, the Back Seat Driver finds the goal of highest priority which is progressable, provided that there is no goal with higher priority with a minimum silence time less than the maximum speech time of the progressable goal. If there is such a goal, it might need to speak before the lower-priority goal had finished, and would need to interrupt it. This algorithm runs goals as soon as possible without wasting time starting goals which will fail. A possible improvement is to add the notion of urgency - an urgent goal is one that must run as soon as possible; an goal which is not urgent can tolerate a slight delay, and need not interrupt a goal of lesser importance.

Time is of the essence for navigation. As the driver travels along the route, the Back Seat Driver looks ahead for the next place where an instruction will be given. It generates some text which describes the instruction (e.g. "At the next set of lights, take a right.") and estimates the time required to speak the text, and then the time when it must begin to speak to allow the driver to carry out the instruction. As the speed of the car changes, this time changes as well. The program usually gives each instruction twice, first 30 seconds in advance and then at the actual time for the instruction. In the latter case, the instruction text is briefer, because it is really only a reminder to the driver, telling where to take the action which has already been described. This spares the driver from needing to look for street signs or landmarks in order to know where to act. This is also an example of using time to refer to space.

When generating the text for an instruction, the Back Seat Driver also looks ahead for the next instruction, and checks whether there will be sufficient time after the current instruction to utter the text for that next instruction. If there is not, as for instance at a "jog" type turn, the text for the second instruction is folded into the text for the first.

The Back Seat Driver has an implementation of the discourse theory of Grosz and Sidner, as in [4]. Space does not allow an adequate description of their theory. Suffice it to say that the program works with DISCOURSE SEGMENTS (which are utterances) and tracks the set of domain concepts and objects (e.g. traffic lights, turns on the route, bridges) which are mentioned in the unfolding discourse. In particular, the program attempts to model the driver's ATTENTIONAL STRUCTURE, which records which concepts are "in focus" at a given moment.

A goal "speaks" by forming a discourse segment which expresses the concept to be conveyed. The segment is converted to a text string for the speech synthesizer, including the necessary synthesizer-dependent escape sequences, and sent to the synthesizer to be spoken. When the synthesizer reports that the string has been spoken, all conceptual items in the segment are added to the current attentional space. This means that the program can henceforth assume the driver is familiar with them. For instance, when giving an instruction for the second time, it is the attentional structure which records that this indeed is the second time the instruction has been given, and thus allows the briefer text.

The attention space allows the program to make more fluent instructions. For example, when the route calls for the driver to make two successive turns in the same direction, the program refers to the second turn as "another" turn, and if the program uses traffic lights as a land mark twice in a row, the second set of lights is also called "another", as in "Now go to another set of lights and take another left.". It allows the program to generate fluent intonation, for example to DEACCENT old information or to raise the pitch range when starting a new topic. In some case, when the attentional model shows that the driver knows everything about the action to be taken except where to do it, the instruction text can be a very brief "Do it now".

No algorithm can guarentee that goals will never be interrupted, since the driver can add new goals at any moment. Should an interruption occur, the synthesizer is halted and the interrupted goal is informed. Since the text is never completely spoken, nothing is added to the attentional space. The goal can handle the interruption by requesting to be rescheduled or by giving up. This latter action is appropriate for goals such as providing help, which can consider themselves to have been satisfied if the user shows that she knows what to do by doing something.

## Conclusions

Speech is inherently temporal. In contrast to graphical systems, programs which engage in conversation must consider the time it takes to say something, and must choose their utterances accordingly. Discourse is concerned with the history of the conversation. Programs which maintain discourse records can generate speech which is more fluent and also more brief, both advantages in spoken language systems. The Back Seat Driver is an implementation of a spoken language system which illustrates these issues.

## Acknowledgments

This work was supported by a grant from NEC Home Electronics Ltd.

# References

[1] Gillian Brown and George Yule. *Discourse Analysis*. Camb Univ Press, 1983.

[2] Herbert H. Clark and Deanna Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22:1–39, 1986.

[3] James R. Davis. *Back Seat Driver: voice assisted automobile navigation*. PhD thesis, Massachusetts Institute of Technology, September 1989.

[4] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[5] Barbara J. Grosz, Karen Sparck-Jones, and Bonnie Lynn Webber, editors. *Readings in Natural Language Processing*. Morgan Kaufman Publishers, Inc., 1986.

[6] Julia Hirschberg, Diane Litman, Janet Pierrehumbert, and Gregory Ward. Intonation and the intentional structure of discourse. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1987.

[7] Brad A Myers. The importance of percent-done progress indicators for computer-human interfaces. In *Proceedings of the Human Factors in Computing Systems Conference*, pages 11–17. ACM, 1985.

[8] Rachel Reichman. *Getting Computers to Talk Like You and Me*. MIT Press, 1985.

[9] Chris Schmandt and Barry Arons. A robust parser and dialog generator for a conversational office system. In *Proceedings of 1986 Conference*, pages 355–365. American Voice I/O Society, 1986.