A Dialog Control Algorithm and Its Performance

Ronnie W. Smith*
Dept. of Computer Science
Duke University
Durham, NC 27706
rws@cs.duke.edu

D. Richard Hipp
Dept. of Computer Science
Duke University
Durham, NC 27706
drh@cs.duke.edu

Alan W. Biermann
Dept. of Computer Science
Duke University
Durham, NC 27706
awb@cs.duke.edu

Abstract

A pragmatic architecture for voice dialog machines aimed at the equipment repair problem has been implemented which exhibits a number of behaviors required for efficient human-machine dialog. These behaviors include:

(1) problem solving to achieve a target goal,
(2) the ability to carry out subdialogs to achieve appropriate subgoals and to pass control arbitrarily from one subdialog to another,
(3) the use of a user model to enable useful verbal exchanges and to inhibit unnecessary ones,
(4) the ability to change initiative from strongly computer controlled to strongly user controlled or somewhere in between, and
(5) the ability to use context dependent expectations to correct speech recognition and track user movement to new subdialogs.

A description of the implemented dialog control algorithm is given; an example shows the fundamental mechanisms for achieving the listed behaviors. The system implementation is described, and results from its performance in 141 problem solving sessions are given.

1 A Voice-Interactive Dialog Machine

A limited vocabulary voice dialog system designed to aid a user in the repair of electronic circuits has been constructed in our laboratory. The system contains a model of the device to be repaired, debugging and repair procedures, voice recognition and sentence processing mechanisms, a user model, language generation capabilities, and a dialog control system which orchestrates the behaviors of the various parts. This paper describes the dialog control algorithm and the performance of the total system in aiding a series of subjects in the repair of an electronic circuit.

*This research was supported by National Science Foundation Grant Number NSF-IRI-88-03802 and by Duke University.

2 Target Behaviors

The purpose of the dialog control algorithm is to direct the activities of the many parts of the system to obtain efficient human-machine dialog. Specifically, it is aimed at achieving the following behaviors.

Convergence to a goal. Efficient dialog requires that each participant understand the purpose of the interaction and have the necessary prerequisites to cooperate in its achievement. This is the *intentional structure* of [Grosz and Sidner, 1986], the goal-oriented mechanism that gives direction to the interaction. The primary required facilities are a problem solver that can deduce the necessary action sequences and a set of subsystems capable of carrying out those sequences.

Subdialogs and effective movement between them. Efficient human dialog is usually segmented into utterance sequences, *subdialogs*, that are individually aimed at achieving relevant subgoals ([Grosz, 1978], [Linde and Goguen, 1978], [Polanyi and Scha, 1983], and [Reichman, 1985]). These are called "segments" by [Grosz and Sidner, 1986] and constitute the *linguistic structure* defined in their paper. The global goal is approached by a series of attempts at subgoals each of which involves a set of interactions, the subdialogs.

An aggressive strategy for global success is to choose the most likely subgoals needed for success and carry out their associated subdialogs. As the system proceeds on a given subdialog, it should always be ready to abruptly drop it if some other subdialog suddenly seems more appropriate. This leads to the fragmented style that so commonly appears in efficient human communication. A subdialog is opened which leads to another, then another, then a jump to a previously opened subdialog, and so forth, in an unpredictable order until the necessary subgoals have been solved for an overall success.

Accounting for user knowledge and abilities. Cooperative problem solving involves maintaining a dynamic profile of user knowledge, termed a *user model.* This concept is described for example in [Kobsa and Wahlster, 1988] and [Kobsa and Wahlster, 1989], [Chin, 1989], [Cohen and Jones, 1989], [Finin, 1989], [Lehman and Carbonell, 1989], [Morik, 1989], and [Paris, 1988]. The user model specifies information needed for efficient interaction with the conversational partner. Its purpose is to indicate what needs to be said to the user to enable the user to function effectively. It also indicates what should be omitted because of existing user knowledge.

Because considerable information is exchanged during the dialog, the user model changes continuously. Mentioned facts are stored in the model as known to the user and are not repeated. Previously unmentioned information may be assumed to be unknown and may be explained as needed. Questions from the user may indicate lack of knowledge and result in the removal of items from the user model.

Change of initiative. A real possibility in a cooperative interaction is that the user's problem solving ability, either on a given subgoal or on the global task, may exceed that of the machine. When this occurs, an efficient interaction requires that the machine yield control so that the more competent partner can lead the way to the fastest possible solution. Thus, the machine must be able to carry out its own problem solving process and direct the actions to task completion or yield to the user's control and respond cooperatively to his or her requests. This is a mixed-initiative dialog as studied by [Kitano and Van Ess-Dykema, 1991], [Novick, 1988], [Whittaker and Stenton, 1988], and [Walker and Whittaker, 1990]. As a pragmatic issue, we have found that at least four initiative *modes* are useful:

(1) *directive* - The computer has complete dialog control. It recommends a subgoal for completion and will use whatever dialog is necessary to obtain the needed item of knowledge related to the subgoal.

(2) *suggestive* - The computer still has dialog control, but not as strongly. The computer will make suggestions about the subgoal to perform next, but it is also willing to change the direction of the dialog according to stated user preferences.

(3) *declarative* - The user has dialog control, but the computer is free to mention relevant, though not required, facts as a response to the user's statements.

(4) *passive* - The user has complete dialog control. The computer responds directly to user questions and passively acknowledges user statements without recommending a subgoal as the next course of action.

Expectation of user input. Since all interactions occur in the context of a current subdialog, the user's input is far more predictable than would be indicated by a general grammar for English. In fact, the current subdialog specifies the *focus* of the interaction, the set of all objects and actions that are locally appropriate. This is the *attentional structure* described by [Grosz and Sidner, 1986] and its most important function in our system is to predict the meaning structures the user is likely to communicate in a response. For illustration, the opening of a chassis cover plate will often evoke comments about the objects behind the cover; the measurement of a voltage is likely to include references to a voltmeter, leads, voltage range, and the locations of measurement points.

The subdialog structure thus provides a set of expected utterances at each point in the conversation and these have two important roles:

(1) The expected utterances provide strong guidance for the speech recognition system so that error correction can be maximized. Where ambiguity arises, recognition can be biased in the direction of meaningful statements in the current context. Earlier researchers who have investigated this insight are [Erman *et al.*, 1980], [Walker, 1978], [Fink and Biermann, 1986], [Mudler and Paulus, 1988], [Carbonell and Pierrel, 1988], [Young *et al.*, 1989], and [Young and Proctor, 1989].

(2) The expected utterances from subdialogs other than the current one can be used to indicate that one of those others is being invoked. Thus, expectations are one of the primary mechanisms needed for tracking the conversation as it jumps from subdialog to subdialog. This is known elsewhere as the *plan recognition problem* and it has received much attention in recent years. See, for example, [Allen, 1983], [Litman and Allen, 1987], [Pollack, 1986], and [Carberry, 1990].

Systems capable of all of the above behaviors are rare as has been observed by [Allen *et al.*, 1989]: "no one knows how to fit all of the pieces together." One of the contributions of the current work is to present an architecture that can provide them all in the limited domain of electric circuit repair. [Allen *et al.*, 1989] describe their own architecture which concentrates on representations for subdialog mechanisms and their interactions with sentence level processing. Our work differs from theirs on many dimensions including our emphasis on achieving mixed-initiative, real time, voice interactive dialog which utilizes a user model.

3 The Zero Level Model

The system implemented in our laboratory (described in great detail in [Smith, 1991]) achieves the above behaviors sufficiently to enable efficient human-machine dialogs in the electric circuit repair environment. The complexity of the system prevents its complete description here. However, a *zero level model* has been devised for pedagogical purposes which illustrates its principles of operation. This model mimicks the mechanism of the dialog machine while omitting the huge array of details necessary to make a real system work. A later section in this paper describes the actual implementation and some of its major modules.

The zero level model is the recursive subroutine ZmodSubdialog shown in figure 1. This routine is entered with a single argument, a goal to be proven, and its actions are to carry out a Prolog-style proof of the goal. A side effect of the proof may be a resort to voice interactions with the user to supply necessary *missing axioms.* In

```
Recursive subdialog routine (enter with a goal to prove)

ZmodSubdialog(Goal)

Create subdialog data structures
While there are rules available which may achieve Goal
  Grab next available rule R from knowledge; unify with Goal
  If R trivially satisfies Goal, return with success
  If R is vocalize(X) then
    Execute verbal output X    (mode)
    Record expectation
    Receive response           (mode)
    Record implicit and explicit meanings for response
    Transfer control depending on which expected response was received
      Successful response:  Return with success
      Negative response: No action
      Confused response:  Modify rule for clarification; prioritize rule for execution
      Interrupt:  Match response to expected response of another subdialog;
                  go to that subdialog        (mode)
  If R is a general rule then
    Store its antecedents
    While there are more antecedents to process Do
      Grab the next one and enter ZmodSubdialog with it
      If the ZmodSubdialog exits with failure then terminate processing on rule R
    If all antecedents of R succeed, return with success

NOTE:  SUCCESSFUL COMPLETION OF THIS ROUTINE DOES NOT NECESSARILY MEAN TRANSFER OF CONTROL
TO THE CALLING ROUTINE.  CONTROL PASSES TO THE SUBDIALOGUE SELECTED BY THE DIALOG CONTROLLER.
```

Figure 1: Zero Level Model

fact, the only voice interactions the system undertakes are those called for by the theorem proving machinery.

The ZmodSubdialog routine has a unique capability needed for proper modeling of subdialog behaviors. Specifically, its actions may be suspended at any time so that control may be passed to another subdialog, another instantiation of ZmodSubdialog that is aimed at achieving another goal. However, control may at a later time return to the current instantiation to continue its execution. In fact, a typical computation involves a set of ZmodSubdialog instantiations all advanced to some point in the proofs of their respective goals; control passes back and forth between them looking for the most likely chances of success until, finally, enough goals are proven to complete the global proof.

The algorithm becomes understandable if an example is followed through in full detail. Assume that the following database of Prolog-like rules are contained in the system knowledge base.

```
General Debugging Rules
  set(knob,Y) <-- find(knob), adjust(knob,Y)
General Dialog Rules
  Y <-- usercan(Y), vocalize(Y)
  vocalize(X)
User Model Rules
  find(knob)
  usercan(adjust(knob,X))
```

Further assume that ZmodSubdialog is entered with argument Goal = set(knob,10). Then in Prolog fashion, the routine grabs the rule R:

$$\text{set(knob,Y)} \leftarrow \text{find(knob),adjust(knob,Y)},$$

and attempts to prove set(knob,10). The algorithm offers three choices depending on whether R is trivial (a single predicate which is not vocalize(X)), R is vocalize(X), or R is a rule with antecedents as is the case here. Thus, in this case the third alternative is followed, and the two antecedents are queued for sequential execution (find(knob) and adjust(knob,10)). Then the first antecedent is selected and ZmodSubdialog is entered with argument Goal = find(knob).

The new subdialog to achieve find(knob) is short, however, since the user model indicates the user already knows how to find the knob because find(knob) exists as an available rule. In fact, ZmodSubdialog finds find(knob) in the rule base, enters the first choice (trivial) and returns with success. If find(knob) had not been found in the user model, the system might have engaged in dialog to achieve this goal. The subdialog control mechanism is not obligated to pass control back to the calling routine, but in this example we will assume that it does.

Satisfactory proof of find(knob) means that the next antecedent at this level, adjust(knob,10), can be attempted, and ZmodSubdialog is entered with this goal. Here, our model selects R = Y $\leftarrow$ usercan(Y), vocalize(Y), unifying Y with adjust(knob,10). Then a deeper call to ZmodSubdialog finds usercan(adjust(knob,X)) in the user model which means control passes to the antecedent vocalize(adjust(knob,10)). This yields another entry to ZmodSubdialog and a selection of the second branch. Here the system executes a voice output to satisfy vocalize(adjust(knob,10)): "Set the knob to one zero."

The handling of the goal find(knob) illustrates how the user model can act to satisfy the theorem proving process and prevent the enunciation of unneeded information. As the theorem proving process proceeds, the fact that a user knows something is represented in the knowledge base as an achieved goal. Theorem proving will encounter this and proceed without voice interaction. In the example, the model already indicates that the user knows how to find the knob so no voice interaction is needed for this goal.

The handling of the goal adjust(knob,10) illustrates how the user model supports the theorem proving process by enabling voice dialog when it is needed. This goal could not be proven by application of rules available in the database and the proof was blocked from further progress. In our terminology, there was a "missing axiom" for the proof. So the system must either give up on this proof or try to fill in the missing axiom by resorting to voice dialog. In the current case, voice dialog was enabled by the user model fact usercan(adjust(knob,X)). This fact opens the way for a query to the user. If the query is positively answered, then the missing axiom is made available.

The role of the user model is thus to supply or fail to supply axioms at the bottom of the proof tree. It both inhibits extraneous verbalism and enables interactions necessary to prove the theorem.

Returning to the example computation of ZmodSubdialog, a voice output has just been given, and the system then records, for this output, the set of expected responses:

```
user(adjust(knob,10))
assertion(knob,position,10)
trivialresponse(affirmation)
trivialresponse(negative)
query(location(knob))
query(color(knob))
```

Expected responses are compiled from the domain knowledge base and from the dialog controller knowledge base.

The user's response is then matched (unified) against the expected meanings and subsequent actions depend on which meaning fits best. Four different types of actions may occur at this point.

(1) The user might respond with some paraphrase of "I set the knob to one zero," or "Okay", which would be interpreted as successful responses. The routine ZmodSubdialog would return with success.

(2) The user might also answer "No" yielding a failure and another cycle around the theorem proving loop looking for an applicable rule.

(3) The user might respond with "What color is the knob?" indicating there may be a chance for a success here if there is further dialog. In fact, our system handles such a need for clarification by dynamically modifying the rule and reexecuting with a newly required clarification subdialog. Here the rule set(knob,10) $\leftarrow$ find(knob), adjust(knob,10) becomes modified to set(knob,10) $\leftarrow$ find(knob), vocalize(knob,white), adjust(knob,10). Reexecution will then yield an explanation of the knob color followed by the previous request reenunciated: "Set the knob to one zero."

(4) The user might respond with an utterance that matches no local expectation. Here the system examines expectations of other subdialogs and searches for an acceptable match. If one is found, control will pass to that subdialog. For example, if the response is, "The LED is flashing seven," and if this is an appropriate response in some other subdialog, control will pass to it.

The control of initiative in ZmodSubdialog is handled by special processing at the steps marked "mode." Thus, in strongly directive mode, verbal outputs will be very positively stated, responses will be expected to be obedient, and interrupts to other subdialogs will be restricted. In less demanding modes, outputs will be weaker or merely suggestive, a wider range of responses will be allowed, and transitions to other subdialogs will be more freely permitted. In the most passive mode, there are few outputs except answers to questions, and an interrupt to any subdialog is acceptable.

A very important part of the dialog system is the *domain processor*, the application dependent portion of the system. It receives all information related to the current problem and suggests debugging steps to carry the process further. We model calls to this processor with the rule: $debug(X) \leftarrow (debuggingstep(X))^*$. This rule is called upon to debug device X with the predicate debug(X) and its effect is to specify a sequence of debugging steps which will be specified dynamically as the problem unfolds and which will lead to repair of the device.

4 The Implementation

4.1 An Integrated Architecture

The implemented system is based on a computational model for dialog processing presented in [Smith, 1991]. The model is applicable to the general class of task-oriented dialogs, dialogs where the computer is assisting the user in completing a task as the dialog ensues. The derived implementation consists of the following modules:

dialog controller - This is the supervisor of the dialog processing system. It provides the subdialog processing highlighted in the zero level model. In addition, it provides the complex algorithm required to properly handle arbitrary clarification subdialogs and interrupts as well as provide the dialog expectations needed to assist with input interpretation. It also maintains all dialog information shared by the other modules and controls their activation.

domain processor - As previously mentioned, it provides suggestions about debugging steps to pursue. In our system the domain processor assists users in circuit repair. It receives user-supplied domain information from the dialog controller and returns suggested debugging subgoals to the controller for consideration. The subgoal selection process weighs the level of expected usefulness of a subgoal with the number of times the subgoal has been previously selected. Consequently, the module may recommend challenging previously reported information if no noticeable progress in the task is being made. In this manner, the module and system can recover from erroneous inputs. If the dialog system is to be used to repair other devices, this is the module that needs to be replaced.

knowledge - This is the repository of information about task-oriented dialogs including: (1) rules for proving completion of goals; (2) user model information including a set of rules for inferring user model information from user input; (3) rules for determining when a clarification subdialog should be initiated; and (4) rules for defining the expectations for the user's response as a function of the type of goal being attempted. Note that *the predefined information of this module is easily modified without requiring changes to the dialog controller.*

theorem prover - This provides the general reasoning capability of the system. In order to allow the dialog controller to control the potentially arbitrary movement among subdialogs, the theorem prover has been made interruptible and put under the supervision of the dialog controller. Consequently, the theorem prover can maintain a set of partially completed proofs, and can activate different proofs as instructed by the dialog controller. It can also dynamically modify the proof structure, a key feature for handling clarification subdialogs. Foremost, the theorem prover is able to suspend itself when it encounters a missing axiom, permitting natural language interaction to assist in axiom acquisition. This contrasts with traditional theorem proving approaches (e.g. Prolog) which simply engage in backtracking when a missing axiom is encountered.

linguistic interface - This consists of the generation and recognition modules. They use information on context and expectation as provided by the dialog controller. The linguistic generator was developed by Robin Gambill. It uses a rule-driven approach that takes as input an utterance specification which encapsulates the desired meaning and produces as output an English string that is sent to a text-to-speech converter for enunciation. Various approaches to generation have been described in [Danlos, 1987], [Hovy, 1988], [Jacobs, 1987], [McKeown, 1985], and [Patten, 1988]. The recognition module was designed to be able to recover from ungrammatical inputs. It will be described in greater detail below.

4.2 Minimum distance parsing

A challenging design problem in any natural language system is the development of a *parser* which will translate the lattice of words output by the speech recognizer into a phrase of some synthetic language (Prolog in our instance) which encapsulates the meaning of what was originally spoken. The design difficulty is exacerbated by the fact that due to speech recognition errors, the word lattice output by the speech recognizer is probably different from what the user spoke, and the fact that users will sometimes speak with a word order or sentence construction which the designers of the parser's grammar did not forsee. (These, and other problems associated with robust parsing of speech are further described in [Eastman and McLean, 1981], [Hayes *et al.*, 1986], and [Young *et al.*, 1989].) In our system, parsing is accomplished using a *minimum-distance parsing* algorithm, similar to algorithms described in [Aho and Peterson, 1972], [Lyon, 1974], and [Levinson, 1985], but extended to accept a lattice as input, instead of a simple word list, and also extended to simultaneously perform *syntax-directed translation* [Aho and Ullman, 1969] into the target language. Minimum-distance parsing finds strings in a given context-free language which can be converted to a string of the input lattice with a minimum number of weighted insertions and deletions. When two or more strings are equidistant from the input, dialog expectation is used to break the tie. The minimum-distance parsing algorithm is very robust – it always finds at least one parse – but it is also computationally expensive. Our algorithm is faster than previously reported parsing algorithms of the same generality, but it still required careful hand-crafting and optimization on a fast machine in order to provide real-time response. It should also be noted that our recognition system is capable of selective verification of meaning. This is highlighted in the following sample dialogs.

5 Sample Dialogs

Figure 2 shows two dialogs obtained from actual usage of the system. These dialogs can be seen in [Hipp and Smith, 1991] that is part of the video presentation of the conference. The words in parentheses found beside certain user statements are the words actually provided by the speech recognizer. Where the discrepancy between the output of the speech recognizer and the words produced by the user is insignificant, they have been omitted.

The first dialog is an entire dialog conducted in declarative mode. Particular points to notice from this user-controlled dialog are:

```
Dialog 1 - Declarative Mode (C denotes the computer, U the user, and R the recognizer)

 1. C: This is the circuit fix it shop. How may I help you?
 2. U: I want to fix this rs111.
 3. C: I am familiar with that circuit.  The LED is supposed to be displaying alternately
       flashing one and seven.
 4. U: The LED is displaying an alternately flashing one and the top corner of a seven.
 5. C: There is supposed to be a wire between connector nine nine and connector one zero zero.
 6. U: It is there.
 7. C: There is supposed to be a wire between connector nine nine and connector one zero two.
 8. U: There is no wire on connector one zero four. (R: stays know wire i connector one zero for)
 9. C: There is supposed to be a wire between connector one zero four and connector one zero two.
    [User adds wire and verifies the circuit is repaired, a three step process.]
10. U: The circuit is working.
11. C: Glad to have been of assistance.  Good-bye.

Dialog 2 - Directive Mode excerpt (begins just after computer describes what circuit does)

 8. U: The LED is displaying nothing.
 9. C: What is the switch at when the LED is off?
10: U: The switch is down.
11. C: Put the switch up.
12. U: Done.
13. C: Put the knob to one zero.
14. U: Which knob?
15. C: Near the top boundary and in the center is a green region.  Now, in the middle
       right of this green region is the control knob.
16. U: The knob is at one zero.
17. C: What is the LED displaying?
18. U: The LED is displaying a one and seven at the same time.
19. C: Put the knob to zero.
20. U: Done.
21. C: What is the LED displaying?
22. U: The same thing.  (R: the same same)
23. C: Did you mean to say the LED is displaying the same thing?
24. U: Yes.
    [Eventually, a missing wire is detected and added, and the computer explicitly guides
    the user through the verification process.  The entire dialog lasts 51 utterances.]
```

Figure 2: Sample Dialogs

(1) Successful processing of anaphoric reference at utterance 6.

(2) Successful shifts to user-initiated subdialogs at utterances 8 and 10.

(3) Successful recovery from speech recognition errors at utterance 8.

The second dialog is an excerpt from a dialog conducted in directive mode (strongly computer-controlled dialog). The total dialog lasted 51 utterances in contrast to the 11 utterance declarative mode dialog. Particular points to notice from this excerpt include:

(1) Computer responses which are more directed and forceful in content than in dialog 1.

(2) Successful handling of a clarification subdialog (utterances 13-16).

(3) Successful verification of the implicit meaning of a user utterance in the presence of speech recognition errors in utterance 22. In contrast with utterance 8 of dialog 1, the system decided an explicit verification subdialog was required to ascertain the meaning of the user's utterance.

6 Experimental Results

The system has been implemented on a Sun 4 workstation with the majority of the code written in Quintus Prolog,[1] and the parser in C. Speech recognition is performed by a Verbex 6000 running on an IBM PC and speech production is performed by a DECtalk[2] DTCO1 text-to-speech converter. The users are restricted to a 125 word vocabulary in the connected speech system.

[1] Quintus Prolog is a trademark of Quintus Computer Systems, Incorporated.

[2] DECtalk is a trademark of Digital Equipment Corporation.

The implemented domain processor has been loaded with a model for a particular experimental circuit assembled on a Radio Shack 160-in-One Electronic Project Kit.

After testing system prototypes with a few volunteers, eight subjects used the system during the formal experimental phase. After a warmup session where the subject trained on the speech recognizer and practiced using the system, each subject participated in two sessions where up to ten problems were attempted. The system ran in declarative mode (user-controlled dialogs) during one session and in directive mode (strongly computer-controlled dialog) in the other session. Subjects attempted a total of 141 dialogs of which 118 or 84% were completed successfully.[3] Subjects spoke a total of 2840 user utterances, with 81.5% correctly interpreted by the system although only 50.0% were correctly recognized word for word by the speech recognizer. The average speech rate was 2.8 sentences per minute, and the average task completion times for successful dialogs were 4.5 and 8.5 minutes, respectively, for declarative and directive modes. The average number of user utterances per successful dialog was 10.7 in declarative mode and 27.6 in directive mode. A detailed description of the experiment and results is given in [Smith, 1991]. The substantially shorter completion times for users in declarative mode can be attributed to the fact that the subjects learned many of the debugging procedures during the experiment and did not need the detailed descriptions given in the directive mode.

7 Summary

A voice interactive dialog architecture has been developed which achieves simultaneously a variety of behaviors believed to be necessary for efficient human-machine dialog. Goal oriented behavior is supplied by the theorem proving paradigm. Subdialogs and movement between them is implemented with an interruptible theorem prover that maintains a set of partially completed proofs and can work on the most appropriate one at any given time. A user model is provided by a continuously changing set of rules that are referenced in the theorem proving process either to enable or inhibit voice dialog. Mixed initiative is made possible by variable types of processing by the output and input routines and by restricting or releasing the ability to interrupt to a new subdialog. Expectation is associated with individual subdialogs, is compiled from domain and dialog information related to each specific output, and is used to improve voice recognition and enable movement between subdialogs.

[3] Of the 23 dialogs which were not completed, 22 were terminated prematurely due to excessive time being spent on the dialog. Misunderstandings due to misrecognition were the cause in 13 of these failures. Misunderstandings due to inadequate grammar coverage occurred in 3 of the failures. In 4 of the failures the subject misconnected a wire. In one failure there was confusion by the subject about when the circuit was working, and in another failure there were problems with the system software. A hardware failure caused termination of the final dialog.

References

[Aho and Peterson, 1972] Alfred V. Aho and Thomas G. Peterson. A minimum distance error-correcting parser for context-free languages. *SIAM Journal on Computation*, 1(4):305–312, 1972.

[Aho and Ullman, 1969] A. V. Aho and J. D. Ullman. Properties of syntax directed translations. *Journal of Computer and System Sciences*, 3(3):319–334, 1969.

[Allen *et al.*, 1989] J. Allen, S. Guez, L. Hoebel, E. Hinkelman, K. Jackson, A. Kyburg, and D. Traum. The discourse system project. Technical Report 317, University of Rochester, November 1989.

[Allen, 1983] J.F. Allen. Recognizing intentions from natural language utterances. In M. Brady and R.C. Berwick, editors, *Computational Models of Discourse*, pages 107–166. MIT Press, Cambridge, Mass., 1983.

[Carberry, 1990] S. Carberry. *Plan Recognition in Natural Language Dialogue*. MIT Press, Cambridge, Mass., 1990.

[Carbonell and Pierrel, 1988] N. Carbonell and J.M. Pierrel. Task-oriented dialogue processing in human-computer voice communication. In H. Niemann, M. Lang, and G. Sagerer, editors, *Recent Advances in Speech Understanding and Dialog Systems*, pages 491–496. Springer-Verlag, New York, 1988.

[Chin, 1989] D.N. Chin. KNOME: Modeling what the user knows in UC. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 74–107. Springer-Verlag, New York, 1989.

[Cohen and Jones, 1989] R. Cohen and M. Jones. Incorporating user models into expert systems for educational diagnosis. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 313–333. Springer-Verlag, New York, 1989.

[Danlos, 1987] L. Danlos. *The Linguistic Basis of Text Generation*. Cambridge University Press, New York, 1987.

[Eastman and McLean, 1981] C. M. Eastman and D. S. McLean. On the need for parsing ill-formed input. *American Journal of Computational Linguistics*, 7(4):257, 1981.

[Erman *et al.*, 1980] L.D. Erman, F. Hayes-Roth, V.R. Lesser, and D.R. Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys*, pages 213–253, June 1980.

[Finin, 1989] T.W. Finin. GUMS: A general user modeling shell. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 411–430. Springer-Verlag, New York, 1989.

[Fink and Biermann, 1986] P.E. Fink and A.W. Biermann. The correction of ill-formed input using history-based expectation with applications to speech understanding. *Computational Linguistics*, 12(1):13–36, 1986.

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

[Grosz, 1978] B.J. Grosz. Discourse analysis. In D.E. Walker, editor, *Understanding Spoken Language*, pages 235–268. North-Holland, New York, 1978.

[Hayes *et al.*, 1986] Philip J. Hayes, Alexander G Hauptmann, Jaime G. Carbonell, and Masaru Tomita. Parsing spoken language: A semantic caseframe approach. In *COLING-86: Proceedings of the 11th International Conference on Computational Linguistics*, pages 587–592, Bonn, August 1986.

[Hipp and Smith, 1991] D. R. Hipp and R. W. Smith. A demonstration of the "circuit fix-it shoppe". A 12 minute videotape available from the authors at Duke University, Durham, NC 27706, August 1991.

[Hovy, 1988] E.H. Hovy. *Generating Natural Language under Pragmatic Constraints.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[Jacobs, 1987] P.S. Jacobs. KING: a knowledge-intensive natural language generator. In Gerard Kempen, editor, *Natural Language Generation*, pages 219–230. Martinus Nijhoff Publishers, Boston, 1987.

[Kitano and Van Ess-Dykema, 1991] H. Kitano and C. Van Ess-Dykema. Toward a plan-based understanding model for mixed-initiative dialogues. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, 1991.

[Kobsa and Wahlster, 1988] A. Kobsa and W. Wahlster, editors. *Special Issue on User Modeling.* MIT Press, Cambridge, Mass., September 1988. A special issue of *Computational Lingusitics.*

[Kobsa and Wahlster, 1989] A. Kobsa and W. Wahlster, editors. *User Models in Dialog Systems.* Springer-Verlag, New York, 1989.

[Lehman and Carbonell, 1989] J.F. Lehman and J.G. Carbonell. Learning the user's language: A step towards automated creation of user models. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 163–194. Springer-Verlag, New York, 1989.

[Levinson, 1985] Stephen E. Levinson. Structural methods in automatic speech recognition. *Proceeding of the IEEE*, 73(11):1625–1650, 1985.

[Linde and Goguen, 1978] C. Linde and J. Goguen. Structure of planning discourse. *J. Social Biol. Struct.*, 1:219–251, 1978.

[Litman and Allen, 1987] D.J. Litman and J.F. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, 11(2):163–200, 1987.

[Lyon, 1974] Gordon Lyon. Syntax-directed least errors analysis for context-free languages. *Communcations of the ACM*, 17(1):3–14, 1974.

[McKeown, 1985] K.R. McKeown. *Text Generation.* Cambridge University Press, New York, 1985.

[Morik, 1989] K. Morik. User models and conversational settings: Modeling the user's wants. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 364–385. Springer-Verlag, New York, 1989.

[Mudler and Paulus, 1988] J. Mudler and E. Paulus. Expectation-based speech recognition. In H. Niemann, M. Lang, and G. Sagerer, editors, *Recent Advances in Speech Understanding and Dialog Systems*, pages 473–477. Springer-Verlag, New York, 1988.

[Novick, 1988] D.G. Novick. *Control of Mixed-Initiative Discourse Through Meta-Locutionary Acts: A Computational Model.* PhD thesis, University of Oregon, 1988.

[Paris, 1988] C.L. Paris. Tailoring object descriptions to a user's level of expertise. *Computational Linguistics*, 14(3):64–78, 1988.

[Patten, 1988] T. Patten. *Systemic Text Generation as Problem Solving.* Cambridge University Press, New York, 1988.

[Polanyi and Scha, 1983] L. Polanyi and R. Scha. On the recursive structure of discourse. In K. Ehlich and H. van Riemsdijk, editors, *Connectedness in Sentence, Discourse and Text*, pages 141–178. Tilburg University, 1983.

[Pollack, 1986] M.E. Pollack. A model of plan inference that distinguishes between the beliefs of actors and observers. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 207–214, 1986.

[Reichman, 1985] R. Reichman. *Getting Computers to Talk Like You and Me.* MIT Press, Cambridge, Mass., 1985.

[Smith, 1991] R.W. Smith. *A Computational Model of Expectation-Driven Mixed-Initiative Dialog Processing.* PhD thesis, Duke University, 1991.

[Walker and Whittaker, 1990] M. Walker and S. Whittaker. Mixed initiative in dialogue: An investigation into discourse segmentation. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 70–78, 1990.

[Walker, 1978] D.E. Walker, editor. *Understanding Spoken Language.* North-Holland, New York, 1978.

[Whittaker and Stenton, 1988] S. Whittaker and P. Stenton. Cues and control in expert-client dialogues. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 123–130, 1988.

[Young and Proctor, 1989] S.J. Young and C.E. Proctor. The design and implementation of dialogue control in voice operated database inquiry systems. *Computer Speech and Language*, 3:329–353, 1989.

[Young *et al.*, 1989] S.R. Young, A.G. Hauptmann, W.H. Ward, E.T. Smith, and P. Werner. High level knowledge sources in usable speech recognition systems. *Communications of the ACM*, pages 183–194, February 1989.