Integration of domain problem solving with natural language dialog: the missing axiom theory

Ronnie W. Smith

Department of Computer Science, Duke University
Durham, NC 27706

ABSTRACT

An important problem in developing natural language dialog systems is to computationally specify when and why the system should speak. This paper proposes interruptible theorem proving as a solution. Theorem proving is used to determine when domain goals are complete. Language is used to acquire missing axioms that may be inhibiting proof completion. The paper describes this "missing axiom" theory for the use of language and how it enables the needed dialog processing behaviors to be achieved. The theory is illustrated with a sample dialog segment obtained from actual usage of an implemented dialog system. Performance results of this system based on over 140 dialogs are also given.

1. WHY SHOULD A COMPUTER SPEAK?

Designers of fully functional natural language (NL) dialog systems must create an architecture that simultaneously achieves a variety of behaviors necessary for efficient human-machine communication. These behaviors include:

- the ability to perform problem solving to achieve a target goal. This requires the ability to integrate the required problem solving capabilities within the application domain with the need to interact via NL dialog.
- the ability to maintain and use information garnered from the dialog.
- the ability to maintain and use a user model to enable useful verbal exchanges and to inhibit unnecessary ones.
- the ability to carry out subdialogs to achieve appropriate subgoals and to pass control arbitrarily from one subdialog to another.

This paper focuses on the problem of integrating domain reasoning with a general NL dialog facility to enable effective and timely assistance for task-oriented dialogs, dialogs held in conjunction with the performance of some task that the computer is helping the user to complete. A solution to this integration problem requires an answer to the fundamental question, "Why should a NL dialog system speak?" Interruptible theorem proving is proposed as the answer. In this paradigm theorem proving is used to determine when required domain actions are completed. When the proof cannot be finished due to missing axioms, language may be used in an attempt to acquire the missing axioms. Thus, the purpose for speaking is to provide the information necessary to: (1) obtain the missing axioms; (2) prove completion of the domain actions; and (3) facilitate completion of the overall task.

This "missing axiom theory" for the use of language enables the required dialog behaviors to be achieved. Furthermore, interruptible theorem proving minimizes the need for time consuming backtracking in the reasoning process, enabling more efficient dialog processing. This is manifested in the implementation of a dialog system based on this theory that provides real-time response and useful task assistance while permitting voice interaction. The paper concludes with a report of results from the usage of the implemented system.

2. STUDYING HUMAN-COMPUTER NL DIALOG INTERACTION

2.1. Previous work

Although all the problems with computational NL understanding and generation have not been solved, researchers have built dialog systems in order to study problems important to dialog processing, such as quantification, presuppositions, ellipsis, anaphoric reference, and user model. Examples of these dialog systems include [4], [6], [7], [10], [13], [14], [15], [16], [25], and [26]. General issues in user modeling are discussed in the collections [17] and [18].

Developing a theory on the role or purpose of language has interested many researchers. As summarized in Allen [3], *speech act* theory was developed based on the realization that statements can do more than just make assertions about the world. The notion of speech acts was first introduced by Austin [5]. Searle [23] and Grice [11] provide further philosophical discussion of the problem. Computational approaches have been proposed in [2], [9], [19], [20], and [21], but these theories have not been tested within an integrated dialog processing system.

2.2. Our research emphasis

These previous research efforts provide important results about isolated factors in NL dialog processing, but they do not study the problem of integrating various processing components into a working, testable system. Allen et al. [1] developed a system which uses a collection of processing modules to perform an integrated discourse analysis, but the computer is not an active participant in the dialog. Furthermore, no dialog processing theory had considered mixed-initiative dialog. This is dialog where: (1) either dialog participant can have control of the dialog; (2) control can change between participants during the dialog; and (3) intermediate levels of control are allowed. In addition, there was a need for a dialog processing theory which yields an implementation whose performance could be measured. Studying the dynamics of human/computer conversational interaction requires the development of a dialog processor that can interact with a user in the same, non-halting way that people interact in conversation.

Consequently, our research goals were to: (1) develop a theory of dialog processing that integrates the various processing components and enables mixed-initiative interaction; and (2) test the theory by building a working system whose performance could be tested and measured via human usage.

A crucial problem in developing this theory was to provide a general, but computationally effective solution to the problem of "Why should the computer speak?" In summarizing the use of belief models and speech act theory for this problem, Allen [3] admits that,"[t]hese techniques have been developed in experimental systems and so far have proved to be too computationally expensive for use in practical systems." The remainder of this paper discusses our solution to this problem.

3. THE ROLE OF LANGUAGE: SUPPLYING MISSING AXIOMS

The first step in integrating the required domain reasoning for providing task assistance with a general NL dialog facility is to include a reasoning system for achieving goals. We chose to use theorem proving, a well understood computational approach. Research into automatic theorem proving has progressed to the point where there now exists at least one practical programming language, Prolog (see Clocksin and Mellish [8] for an introduction to the language), based on the theorem proving paradigm.

Our theory then utilizes language interaction with the user to acquire the missing axioms needed for proving a target goal. Consider the proof tree of figure 1. a is proven by proving b, c, and d; and b is proven by proving $b1$ and $b2$. Suppose that the process of proving a yields subgoal b which leads to subgoals $b1$ and $b2$. Further suppose that $b1$ and $b2$ are domain facts not ordinarily available to the machine. Our system resorts to natural language in this case and interacts with the user to try to verify them. For example, $b1$ might require that the user know how to locate an item and $b2$ might require the user to report on that item's status. These are known as "missing axioms" and could yield the following dialog (C denotes the computer and U the user):

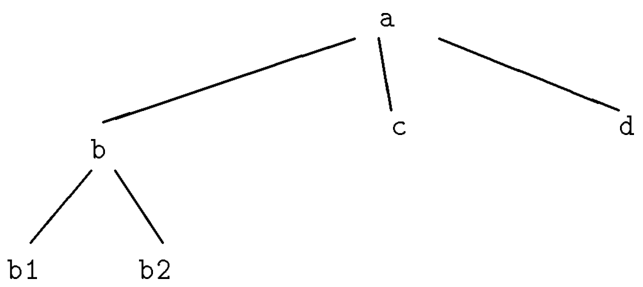

Figure 1: Sample theorem description

```
C: The knob is at the top of the green region.
U: I see it.
C: What is its setting?
U: Zero.
```

Completion of this dialog enables the proof of b and possibly, at a later time a.

This example also illustrates how user modeling is done. For example, if an earlier interaction had involved successfully referencing the knob, then $b1$ would be included as an achieved fact in the user model. The proof of b would have achieved $b1$ trivially and omitted the first comment about locating the knob.

Furthermore, a close correlation between subdialogs and domain actions is obtained. A subdialog consists of all the language interaction pertaining to a domain action. Movement between and within subdialogs is based on determining the domain action to which a user input is relevant. This requires handling the situation that a person's response does not satisfy a missing axiom. This situation is discussed in the following section.

4. DYNAMIC PROOF MODIFICATION

Consider the following possible computer utterance, "Please turn the switch up." This might be uttered in the hope of obtaining the missing axiom that the switch is up. However, if the user responds with, "Where is the switch?", or "I do not see the switch", then the utterance does not satisfy the missing axiom. In the normal theorem proving paradigm, this would cause backtracking to find another rule that could be used for the proof. This would require explicitly listing theorem descriptions for each possible response that provides an axiom other than the one needed, a computationally unwieldy and non-practical approach. This type of clarification subdialog is very common to general conversation, especially task-oriented dialogs, and requires a general solution.

Our solution requires the dynamic modification of the original theorem to insert completion of the subaction, in this case, learning the location of the switch, so that it must be completed before the original action may be completed. In order for this solution to be viable, the theorem proving process must be under the guidance of an overall controlling process for the dialog processor (called the *dialog controller*) that must decide when theorem proving is to be activated, when language can be used to acquire missing axioms, and when theorems must be dynamically modified. This has led to the development of the Interruptible Prolog SIMulator (IPSIM), a Prolog-like theorem prover which can be accessed by a standard set of commands usable by the dialog controller. The utility of interruptible theorem proving will be illustrated via the following example.

5. SAMPLE DIALOG SEGMENT

We will discuss how the integration of domain reasoning with dialog processing via interruptible theorem proving occurs within the context of the following dialog segment obtained from usage of our implemented system.

```
 (1) C: Is there a wire between connector 84 and connector 99?
 (2) U: It is not there.
 (3) C: Add a wire between connector 84 and connector 99.
 (4) U: I need help.
 (5) C: Obtain a wire.
 (6) U: Done.
 (7) C: Connect metal of one free end of the wire to connector 84.
 (8) U: How?
 (9) C: Bend back connector 84.
(10) U: Done.
(11) C: Insert free end of wire into connector 84.
(12) U: Wire between 84 and 99 is connecting.
(13) C: What is the LED displaying?
```

Figure 2 illustrates the relationship between the theorem-proving process and the subdialogs. After determining that a wire between connectors 84 and 99 is missing (MAIN GOAL 1), the computer initiates a subdialog to have this wire added (MAIN GOAL 2). Utterance 4 initiates a clarification subdialog for this main goal. Consequently, the dialog controller instructs IPSIM to insert into the active theorem specification the subgoal on learning how to accomplish the main goal (the "learn to do add" subgoal). The insertion occurs immediately before the theorem step of finding or acquiring the needed axiom about adding the wire.

Resumption of theorem proving leads to an expansion of the subgoal on learning how to add the wire. This expansion requires leading the user through each substep of the main goal. The first two substeps are to locate connectors 84 and 99. These are satisfied trivially by the theorem prover without the use of language. The user model axioms indicating that the user knows how to locate these connectors were inferred from the user statement, "It is not there."

The next substep requires language to acquire the missing axiom (utterances 5 and 6). The subgoal introduced by utterance 7 (connect(end1,84)) requires initiation of a second clarification subdialog for learning how to do its substeps as a result of utterance 8. After completing the first two of these substeps, utterance 12 indicates the wire has been added. The system searches through the possibilities for missing axioms for the unsatisfied goals introduced at utterances 11, 7, and 3 to determine that this supplied the missing axiom for the main goal, adding the wire. Consequently, proofs of the substeps are discontinued (note also that the proof of the subgoal "connect(end2,91)" does not even need to be started), and the dialog continues with the next main goal (MAIN GOAL 3).

This example has shown that in addition to achieving the desired integration of domain reasoning with dialog processing, the missing axiom theory for the role of language yields solutions for the following dialog processing problems:

1. construction of a discourse model - Integration of theorems, the utterances relevant to these theorems, and the expectations for responses that supply missing axioms yields a constructive method for creating a discourse model first proposed by Grosz and Sidner [12], but for which they did not offer a method for constructing the model dynamically during the course of a dialog.

2. use of user model - The user model is maintained as a set of axioms acquired from inferences based on user input. The axioms may then be used by the theorem prover in determining action completion to

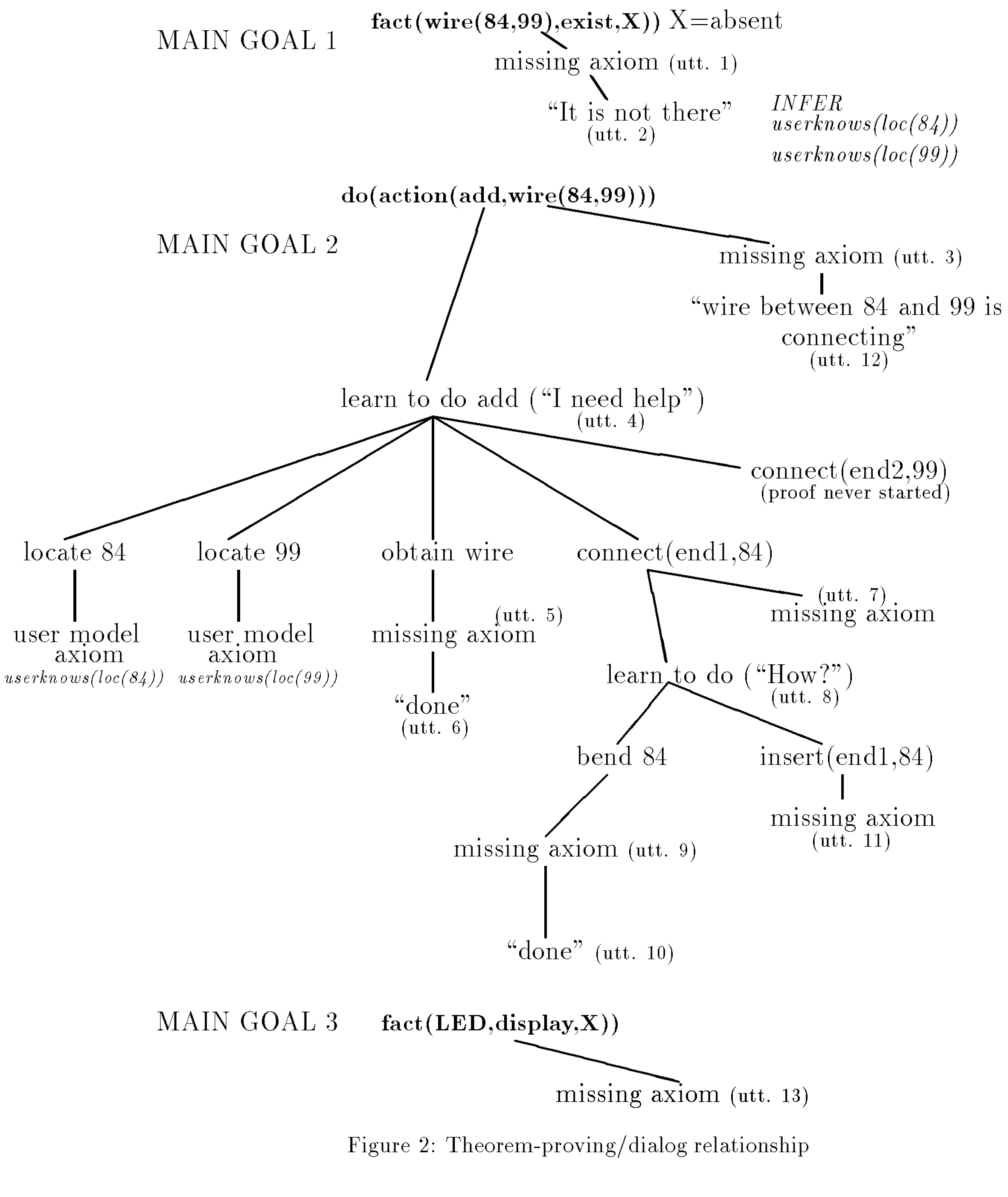

Figure 2: Theorem-proving/dialog relationship

avoid telling the user information already known such as the location of the connectors in the sample dialog segment.

3. handling subdialog movement - Based on the determination of the theorem subgoal to which the user utterance is relevant, the dialog controller can order the theorem prover to move to the appropriate portion of the theorem. Furthermore, the same mechanism works for movement between different subdialogs as well.

6. IMPLEMENTATION AND EVALUATION

The working dialog 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. Separate modules were developed to

- provide overall control for dialog processing (the *dialog controller*)
- provide domain problem solving (the *domain processor*)
- provide the following general knowledge about task oriented dialogs: (the *knowledge* module)
  — rules for proving completion of goals;
  — user model information including a set of rules for inferring user model information from user input; and
  — rules for determining when a clarification subdialog should be initiated
- provide a general reasoning capability (the *IPSIM theorem prover*)
- provide actual linguistic capabilities (the *parser* and *generator*)

It should be noted that the dialog controller and knowledge module are based on general principles of task-oriented dialogs described in Smith [24]. Domain specific values are provided by the particular domain processor being used. The implemented domain processor provides information about repairing a Radio Shack 160 in One Electronic Project Kit. The particular circuit being used causes the Light-Emitting Diode (LED) to alternately display a one and seven. The system can detect errors caused by missing wires as well as a dead battery.

For the dialog system, 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.

After testing system prototypes with a few volunteers, eight subjects used the system during the formal experimental phase. After a warmup session where a subject trained on the speech recognizer and practiced using the system, each subject participated in two sessions where up to ten problems were attempted. Subjects attempted a total of 141 dialogs of which 118 or 84% were completed successfully. 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. Subjects spoke a total of 2840 user utterances, with an average speech rate of 2.8 sentences per minute. The average task completion time for successful dialogs was 6.5 minutes. Detailed results are reported in Smith [24].

[1] Quintus Prolog is a trademark of Quintus Computer Systems, Incorporated.
[2] DECtalk is a trademark of Digital Equipment Corporation.

7. SUMMARY

This paper has presented the paradigm of interruptible theorem proving as a means for handling the problem of integrating domain problem solving with NL dialog to facilitate construction of NL dialog systems for interacting with human users to provide task assistance. This approach requires that theorem proving be used to determine if task goals have been completed and that language be used to acquire needed missing axioms. Advantages we have observed with this approach include:

- The language-directed nature of the theorem proving process minimizes backtracking, greatly enhancing efficiency. Whenever a missing axiom is detected, language is used, and the depth-first proof can be continued. When several missing axioms are active, the proof terminates when the missing axiom for the top level goal is acquired. The subgoal proofs can simply be discontinued.
- The paradigm facilitates solutions to other important dialog processing problems including: (1) maintaining and using a model of the dialog; (2) maintaining and using a model of the user; and (3) handling movement between different subdialogs.
- The paradigm facilitates the construction of NL dialog systems whose performance can be tested with human users. Our formal testing involving over 140 dialogs and 2800 utterances is the most extensive formal testing of which we are aware.

An important unresolved issue concerns theorem failures. A theorem failure occurs when a proof cannot be completed, even after language is used. We have identified three sources for theorem failures:

1. Miscommunication between the computer and user. In this case, theorem failure can be handled by reattempting the goal, but by trying to complete each substep (i.e. go into more detail in proving goal completion).
2. A physical breakdown occurs. For example, "the switch is stuck," may cause a theorem failure in proving that the switch has been set to a certain position. In this situation, the goal cannot be completed. This could be reported to the domain processor who can then suggest another goal.
3. An occurrence of a situation outside the system's knowledge. In the implemented system, this occurred four times when subjects misconnected wires. In each case, the resulting LED display was unknown to the system, causing it to be unable to provide assistance. This is a problem endemic to knowledge-based systems and the subject of active research in learning and common sense reasoning. Some type of learning procedure in conjunction with an extension of a "question-asking" mechanism such as that proposed by Ram [22] might provide a solution.

In practice, theorem failures were very infrequent, but further study is needed on this problem to further improve the robustness of systems based on the missing axiom theory of language use for NL dialog systems.

8. ACKNOWLEDGMENTS

This research was supported by National Science Foundation Grant Number NSF-IRI-88-03802. Other researchers who have contributed to this project include Alan W. Biermann, Robert D. Rodman, Ruth S. Day, D. Richard Hipp, Dania Egedi, and Robin Gambill. The author also wishes to express his gratitude to the anonymous referees, whose comments were very helpful in the preparation of the final paper.

References

[1] 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.

[2] J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178, 1980.

[3] J.F. Allen. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1987.

[4] J.F. Allen, A.M. Frisch, and D.J. Litman. ARGOT: The Rochester dialogue system. In *Proceedings of the 2nd National Conference on Artificial Intelligence*, pages 66–70, 1982.

[5] J.L. Austin. *How to Do Things with Words*. Harvard University Press, Cambridge, Mass., 2nd edition, 1962. Edited by J.O. Urmson and M. Sabisà.

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

[7] S. Carberry. Modeling the user's plans and goals. *Computational Linguistics*, 14(3):23–37, 1988.

[8] W. F. Clocksin and C. H. Mellish. *Programming in Prolog*. Springer-Verlag, Berlin, third edition, 1987.

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

[10] R.E. Frederking. *Integrated Natural Language Dialogue: A Computational Model*. Kluwer Academic Publishers, Boston, 1988.

[11] H. P. Grice. Logic and conversation. In P. Cole and J.L. Morgan, editors, *Syntax and Semantics, Vol. 3: Speech Acts*, pages 41–58. Academic Press, New York, 1975.

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

[13] C.D. Hafner and K. Godden. Portability of syntax and semantics in Datalog. *ACM Transactions on Office Information Systems*, pages 141–164, April 1985.

[14] G. G. Hendrix, E.D. Sacerdoti, D. Sagalowicz, and J. Slocum. Developing a natural language interface to complex data. *ACM Transactions on Database Systems*, pages 105–147, June 1978.

[15] W. Hoeppner, T. Christaller, H. Marburger, K. Morik, B. Nebel, M. O'Leary, and W. Wahlster. Beyond domain-independence: Experience with the development of a German language access system to highly diverse background systems. In *Proceedings of the 8th International Joint Conference on Aritificial Intelligence*, pages 588–594, 1983.

[16] S.J. Kaplan. Cooperative responses from a portable natural language query system. *Artificial Intelligence*, 19(2):165–187, 1982.

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

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

[19] D.J. Litman and J.F. Allen. Discourse processing and commonsense plans. In P. R. Cohen, J. Morgan, and M.E. Pollack, editors, *Intentions in Communication*, pages 365–388. MIT Press, Cambridge, Mass., 1990.

[20] M. E. Pollack. Plans as complex mental attitudes. In P. R. Cohen, J. Morgan, and M.E. Pollack, editors, *Intentions in Communication*, pages 77–103. MIT Press, Cambridge, Mass., 1990.

[21] R. Power. The organisation of purposeful dialogues. *Linguistics*, 17:107–151, 1979.

[22] A. Ram. AQUA: asking questions and understanding answers. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 312–316, 1987.

[23] J. R. Searle. Indirect speech acts. In P. Cole and J.L. Morgan, editors, *Syntax and Semantics, Vol. 3: Speech Acts*, pages 59–82. Academic Press, New York, 1975.

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

[25] D. L. Waltz. An english language question answering system for a large relational database. *Communications of the ACM*, pages 526–539, July 1978.

[26] R. Wilensky, D.N. Chin, M. Luria, J. Martin, J. Mayfield, and D. Wu. The Berkeley UNIX consultant project. *Computational Linguistics*, 14(4):35–84, 1988.