Resolving Anaphora in a Portable Natural Language Front End to Databases

Flávia A. Barros and **Anne DeRoeck**
Department of Computer Science
University of Essex
Colchester - CO4 3SQ - U.K.
`[barrf - deroe]@essex.ac.uk`

Abstract

An analysis of the evolution of Natural Language front ends in the last three decades shows that the growth in portability brought, as a side effect, the narrowing of the provided coverage of contextually based linguistic phenomena, such as anaphora and ellipsis.

This paper presents the design and state of development of a computational mechanism which provides pronominal Anaphora Resolution within the environment of a highly portable Natural Language front end to databases, SQUIRREL.[1] Simple cases of Ellipsis are also treated by the proposed model.

An Overview of SQUIRREL is presented, followed by a description of the *Discourse Module* and the results achieved so far. The prototype is implemented in C-Prolog.

1 Introduction

The development of Natural Language (NL) systems for data retrieval has been a central issue in NL Processing research for the last three decades, motivated by the aim of helping non-expert database users. When we try to draw a line of evolution of such systems, it can be observed that growth in portability, essential for commercial viability, came at a cost in terms of broader linguistic coverage.[2]

Earlier systems, mostly research motivated, were mainly developed for a single application, using domain-dependent information for treating contextual phenomena (*eg*, DEACON (Craig *et al.*, 1966), SHRDLU (Winograd, 1972), LUNAR (Woods, 1973), LADDER (Hendrix *et al.*, 1978)). In contrast, the subsequent generation of interfaces carried a higher emphasis on portability in their design (*eg*, INTELLECT (Harris, 1984), IRUS (Bates *et al.*, 1986), TEAM (Grosz *et al.*, 1987)). These systems, however, offer a reduced coverage of discourse phenomena, a central issue when continuity in the database consultation carries some priority. Thus, the ideal NL Front End (NLFE) should carry a broader linguistic coverage, in order to support a user focused query process, combined with a high degree of portability.

In this light, we designed a *Discourse Module*, which is incorporated into a highly portable NLFE, SQUIRREL (DeRoeck *et al.*, 1991). The system was originally conceived with a single-query based mode of consultation. By providing for anaphora and simple cases of ellipsis resolution, the *Discourse Module* yields continuous consultations without the use of world models (to maintain the system's general portability).

2 Issues in Anaphora Resolution

Our primary goal is the achievement of dialogue-like querying by extending SQUIRREL to a system capable of dealing with basic pronominal anaphora and ellipsis. Information about each query is made available to the following queries, such that references to entities already introduced can be resolved.

This solution is common practice among NLFEs implementations (*eg*, LDC-1 (Ballard *et al.*, 1984), Datenbank-DIALOG (Trost *et al.*, 1988)). However, it is subject to limitations. In particular, sequences like the following cannot be handled:

`Query:` *who works for which salary in shoes?*
`DB answer:` *[malcolm - $5,000.00]*

`Query:` *who is* **his** *boss?*

because the resulting query

`Query:` *who is the boss of* **[who works for which salary in shoes]** *?*

leads the system into a type error, as a personal pronoun was substituted by a sentence. This was our

[1]The current system forms the base line for a joint SERC/DTI funded collaborative project between the University of Essex and Status IQ Ltd. for constructing an integrated platform for the retrieval of structured and textual data through Natural Language queries.

[2]See (Barros and DeRoeck, 1993) for a comprehensive review on Portable NL front ends.

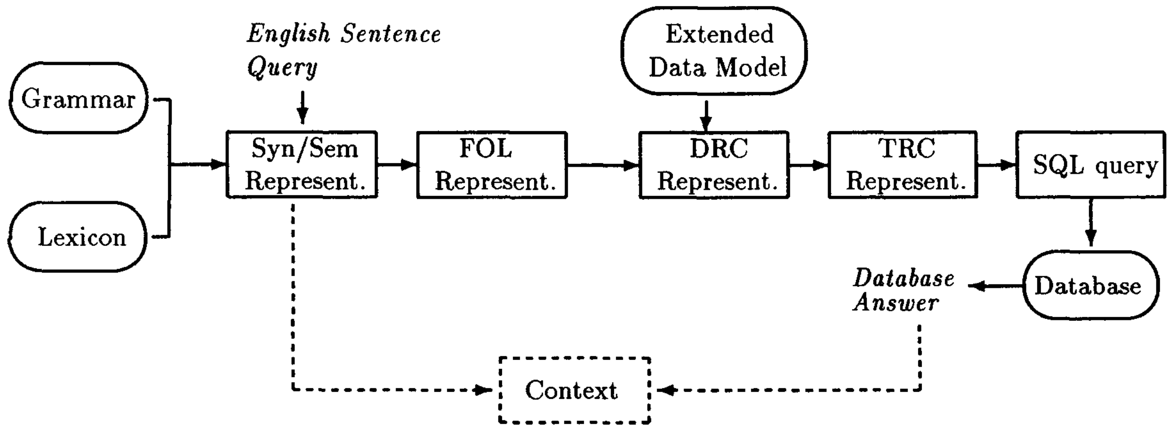

Figure 1: *SQUIRREL with Discourse Module.*

main motivation for keeping information conveyed by the DB answer, so it could be used for future reference. We consider this essential for achieving a dialogue-like mode of consultation.

3 Overview of SQUIRREL

The system consists of a portable Natural Language front end prototype for the interrogation of logical and relational database systems (DeRoeck *et al.*, 1991). It is divided into two main sections: the *Front End* and the *Back End* [Fig. 1].

The *Front End* takes the input sentence, producing syntactic and semantic representations, which it maps into First Order Logic. All representations are independent of the domain of application or database model. Syntactic and semantic rules work in tandem. The former are a feature-based CFG, whereas the latter are expressed in Property Theory (Turner, 1987). The lexicon is incomplete, treating unknown words as proper nouns to be fully interpreted when reaching the database.

The *Back End* uses an Extended Data Model to map the logical representation into expressions in the Domain Relational Calculus (DRC), which is translated via Tuple Relational Calculus (TRC) into SQL (a standard query language) by means of a syntactic transducer. All representations at this level are domain dependent.

Both the *Front End* and *Back End* were designed to guarantee modularity and portability to the system as a whole. Strict separation between domain dependent and independent components must be maintained for the sake of portability. The system has no world model embedded in it, and no inference engine. Only the lexicon and the table-based Extended Data Model have to be customised for a new domain of application.

SQUIRREL maintains three levels of ambiguity, induced by the syntax, the semantics, and the domain. The *Back End* has a type checker, which uses the Extended Data Model to resolve ambiguity from the semantic level. At each level, all possible representations are generated by the system, and tried one at a time. Only the appropriate ones survive the type checking and the database consultation.

As a consequence, more than one successful answer can be obtained from the same query. All successful answers are presented to the user, who is in charge of choosing one. This feature was added to the original system in order to give the user control over which elements are added to the *context* during the consultation.

4 The Discourse Module - Overview

The core of the *Discourse Module* is the *context*, a dynamic list of candidate antecedents for anaphoric reference. The *context* grows as a stack, i.e., candidates selected from each query and its DB answer are stored on top of the candidates from the previous queries as the consultation evolves. All candidates are represented in the same format.

Selection of candidates from the query is regulated by rules embedded in the system's grammar, where each syntactic rule has its associated context rule. Entries are associated with information pertaining to category, number, gender and semantic representation. Since the lexicon allows open classes (such as proper names, for which no specific lexical entry exists), some of this information may not become available until the query reaches the database. At this point, a separate database interrogation will supply gender information for proper nouns in the context.

Selecting which items in database answers are added to the *context* is less straightforward, as no syntactic or semantic information is available concerning a particular answer. Furthermore, the selection may depend on a specific application and should be a factor for customisation. As a consequence, information on which entities are to be kept as candidates for reference in the *context* is encoded in the

Extended Data Model. Entries are formatted and associated with syntactic and semantic information (on the basis of the characteristics of the database domain from which they are retrieved) and, in cases where the data derives from domains associated with proper nouns, gender is retrieved immediately.

This *context* grows dynamically and is passed from query to query. Nevertheless, this structure does not grow indefinitely during the consultation (the *context* updating mechanism is presented in §5.2).

When an anaphor is encountered in a query, a candidate is chosen among the available possible antecedents, and its semantics is inserted in the query's semantic representation, which is then passed to the back end in the normal way. The binding mechanism is presented in §5.3.

Clearly, much hinges on an effective process to determine appropriate antecedents. In the context of this application, which strongly emphasizes portability and, hence, seeks to avoid incorporating general world knowledge, this issue is subject to constraints.

5 The Context

5.1 Rationale

In resolving anaphoric reference in NLFEs to databases, due respect must be given to user requirements; it must remain clear at all times which query has been answered. Findings in Anick *et al.* (1991) also apply here. The operation of the front end must be clear to the user, who must retain the ability to affect the system's decisions. As a consequence, we adopt a protocol whereby *(i)* the user can always reject the bindings offered by the system, and *(ii)* choice between competing candidates is in the hands of the user.

This scenario has some consequences. First of all, our strategy aims not to completely resolve an anaphoric reference at all costs, but to present the user with alternatives selected on the basis of reliable system information. Secondly, to make this process helpful, in a manageable way, the choice of candidates must be focused, intuitively credible, and of limited size.

5.2 Context Structuring

Helpful selection of candidate antecedents presupposes a sensitivity to the structure of the current discourse. More is needed than a simple collection of items based on compliance with syntactic constraints. The literature offers a collection of approaches to modelling discourse structure.

Some views concentrate on deriving coherence relations between discourse segments, with the help of world models (Hobbs, 1979; Reichman, 1984). Work on Discourse Structure Theory (Grosz, 1977; Grosz and Sidner, 1986) searches for automatic ways of segmenting discourse based on intentions and purposes embedded in discourse segments.

Most of the results available are not readily adaptable to the current type of application. No world model can be introduced without severe consequences for portability. Segmentation information is not available and cannot be realised in advance since consultation is on-line.

The lack of clues regarding how to segment the dialogue between user and interface, and how to identify the relationships between such segments restricts the possible solutions. Nonetheless, some domain information is present in the Data Model and can be exploited. The segmentation process deployed here relies on two potential sources for identifying coherence in relational queries. First of all, 'meaningful' queries are always associated with a complete access path covering relations and attributes in the database: if we represent the data model as a graph, a meaningful query will always cover a connected subgraph. This gives us a measure of cohesion within a single query.

Building on this, we can develop a notion of discourse domain covered by successive queries by comparing the access paths involved in them. To use the graphical analogy as above, a collection of queries covers a discourse domain if their graphs intersect. Finally, the degree and area of intersection (consulted attributes) may offer information which can be used in the identification of *focus*.

Candidates for anaphoric reference are grouped in segments, each containing all successive queries which share part of an access path. The first query starts the first segment of the *context*. When a new query is entered, its covered domain is matched against that of the segment on top of the *context*. If the intersection is not empty, candidates from the query are added to this segment. In case the intersection is empty, the system identifies a change of *focus* on the consultation, and a new segment is started. In order to allow the user to return to the previous *topic* after the change of *focus* occurred, a number of segments are held in the *context*. This number can vary from application to application, and the current limit is set to three.

Following Grosz and Sidner (1986), segments occur in sequence, or are embedded, to allow users to elaborate on a change of *focus* before returning to the previous topic. In case the current segment intersects with the second most recent one on the *context* list (if any), this can be seen as a return to the previous topic (segments 1 and 3 in Fig 2). The current segment will continue to grow independently, but the candidates in the second most recent segment will become available for reference.

Within a segment, candidates are grouped by query number. When a candidate re-occurs, it is placed on the top of the *context* list, and its previous occurrence is deleted, regardless what segment it belongs to. The antecedent of a resolved anaphor is also added to the top [Fig. 2]. This strategy allows for the representation of a notion of 'distance' between candidate antecedents and anaphor.

5.3 The Binding Mechanism

When an anaphoric expression is encountered, all candidates in the current segment with appropriate syntactic characteristics are selected and placed in the *foci list* (Sidner, 1983). This list is presented to the user, who must select a candidate or reject all options (in case there is more than one) [Fig. 3].

Once a candidate is selected, its semantic representation is spliced into the First Order Logic representation of the current query, and the normal querying process is resumed.

5.4 Examples

Example 1: Context updating mechanism [Fig. 2]

```
query: who is edna's boss?
db answer: [malcolm]

query: who supplies shoes?
db answer: [peter&CO]

query: what is sylvia's salary?
db answer: [2500]

query: who is her boss?
** USER: Please choose one substitute for
the pronoun 'her':
1 - sylvia
2 - none above              number: 1
db answer: [edna]

query: what is kate's address?
db answer: [spring ave]

query: what is her account?
** USER: Please choose one substitute for
the pronoun 'her':
1 - kate
2 - none above              number: 1
db answer: [678.654]
```

Candidates are grouped by query number and segment number. In the fourth query above, only *sylvia* is presented as a substitute for the anaphor *her*, since this is the only entry with appropriate syntactic features in the current segment (segment 3 - Fig. 2). In case the user rejects it, *edna* (second most recent segment) will be presented as a second option. Similarly, in the last query, only *kate* is presented as an initial choice (current segment).

Example 2: The binding mechanism [Fig. 3]

```
query: who is edna's boss?
db answer: [malcolm]

query: who is sylvia's boss?
db answer: [edna]

query: who works for her?
** USER: Please choose one substitute for
the pronoun 'her':
1 - sylvia
2 - edna
3 - none above              number: 2
db answer: [mary, sylvia, ted]
```

The binding mechanism relies solely on information provided by the Data Model, since there is no world model available. The absence of such a knowledge base is justified by the preoccupation with portability. However, the task of dealing with discourse phenomena is made more difficult.

The user is given the burden of establishing priority when chosing candidates for anaphora. In **Example 2**, for instance, the system has no means of disambiguating between the two possible candidates for binding the pronoun 'her' (*sylvia, edna*), since both have the same properties. Note that humans would not be able to select one either.

Care must be taken in using information about candidates to resolve ambiguity (for instance, the fact that *edna* is a boss, whereas *sylvia* is not), since this could lead into erroneous interpretations. The person *edna* can be used in different contexts within the same dialogue, although it was introduced in the *context* via a query where she appears as boss. Imagine that she is a boss in a shop, but also a registered customer. In such case, references to her name as a customer would be disregarded.

6 Plurals

The importance of a proper context updating mechanism is better seen when we focus on the treatment of plurals. Currently, the system is being extended to cope with plural nouns and groups, referred to by pronouns like *they, them, their.* The incorporation in the *context* of these elements appearing in the query or DB answer is processed as follows:

(a) plural nouns appearing in the query or DB answer are kept as plural elements, having one entry in the *context*;

(b) groups resulting from a DB answer have each of their elements incorporated in isolation, as a singular noun, as well as one entry with all elements combined as a group element;

(c) conjunctions appearing in the query are treated as in **(b)**.

In the present system, problems mostly concern the identification of which elements, appearing in *separate queries* or in a *query/DB answer*, should be gathered together to constitute a group entry.

When a plural anaphor is encountered, the *context* is searched. In case there are no plural/group candidates available, or the user rejects them all, elements in the current *discourse segment* will be gathered together and presented to the user as a group.

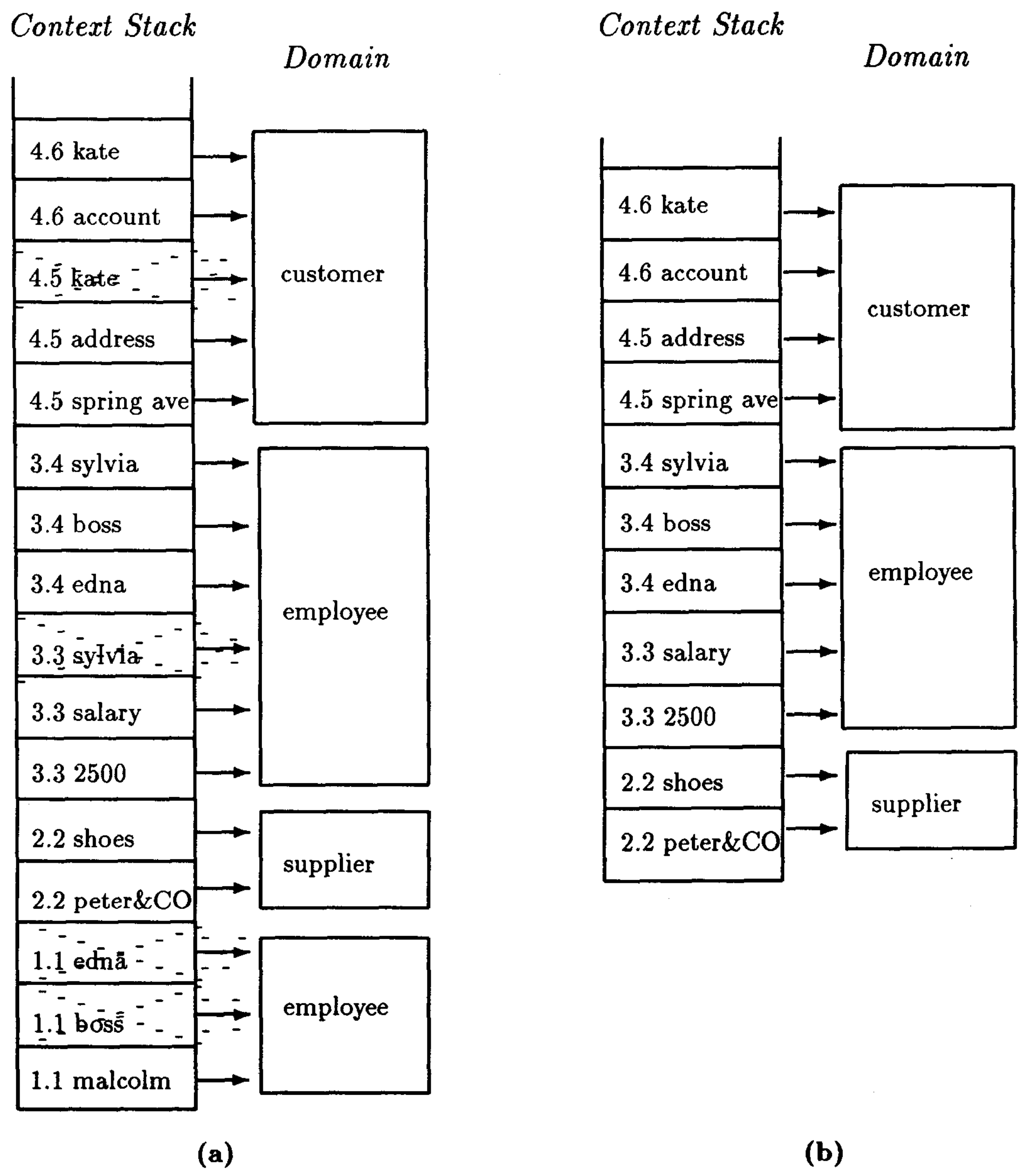

Figure 2: *Context Behaviour*

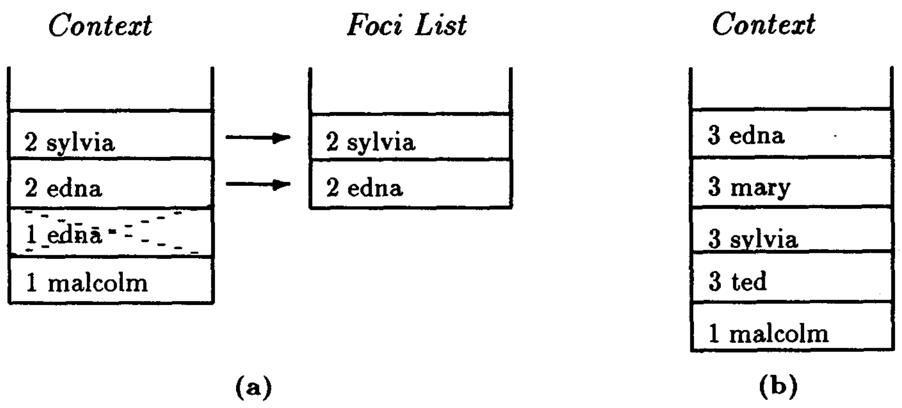

Figure 3: *Context - Foci List*

7 Conclusions

We presented here a module for anaphora resolution in a highly portable NLFE - SQUIRREL, which allows for continuous consultations whilst maintaining the system's portability.

A mechanism to deal with possible alternative successful DB answers has also been added. Such features did not constitute a problem for the original system, since no information was passed forward. With the incorporation of answers into the context, it became necessary to allow users to choose among the multiple possibilities presented by the system, assuring that a unique answer is selected.

We treat some simple cases of *there* (although it is not an anaphor, but a deictic adverb), due to the high rate of usage of such pointing back device in dialogues. In the domain covered by this implementation, its use allows reference to addresses and locations like department and floor.

We maintain that portability is an important property, as NLFEs to databases only make sense in a commercial context. We have demonstrated that it is possible to include reliable, user-oriented features of discourse phenomena in the coverage of modular NLFEs without recourse to world models, safeguarding portability.

References

Anick, Peter G.; Brennan, Jeffrey D.; Flynn, Rex A.; Hanssen, David R.; Alvey, Bryan; and Robbins, Jeffrey M. (1990). "A Direct Manipulation Interface for Boolean Information Retrieval via Natural Language Query." In *13th International Conference on Research and Development in Information Retrieval (ACM-SIGIR)*, 135-150. Brussels.

Ballard, Bruce W.; Lusth, John C.; and Tinkham, Nancy L. (1984). "LDC-1: A Transportable, Knowledge-Based Natural Language Processor for Office Environments." In *ACM Transactions on Office Information Systems* 2(1), 1-25.

Barros, Flavia A., and DeRoeck, Anne (1993). "Portable Natural Language Front Ends - A Review". Research Report CSM-194. Dept. of Computer Science. University of Essex, U.K.

Bates, Madeleine; Moser, M.G.; and Stallar, David (1986). "The IRUS Transportable Natural Language Database Interface." In *Expert Database Systems*, edited by Larry Kerschberg, 617-630. The Benjamin/Cummings Pub. Co. Inc.

DeRoeck, Anne; Fox, Chris; Lowden, Barryl; Turner, Ray; and Walls, Bryan (1991). "A Natural Language System Based on Formal Semantics." *Proceedings of the International Conference on Current Issues in Computational Linguistics*, 221-234. Penang, Malaysia.

Grosz, Barbara J. (1977). "The Representation and Use of Focus in a System for Understanding Dialogs." In *Readings in Natural Language Processing*, edited by Barbara J. Grosz, Karen S. Jones, and Bonny L. Webber (1986), 353-362. Morgan Kaufmann Pub. Inc.

Grosz, Barbara J., and Sidner, Candace L. (1986). "Attention, Intention, and the Structure of Discourse." In *Computational Linguistics* 12(3), 175-204.

Grosz, Barbara J.; Appelt, Douglas E.; Martin, Paul A.; and Pereira, Fernando C.N. (1987). "TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces." In *Artificial Intelligence* 32, 173-243.

Harris, Larry R. (1984). "Experience with INTELLECT: Artificial Intelligence Technology Transfer." In *The AI Magazine* 2(2), 43-50.

Hendrix, Gary G.; Sacerdoti, Earl D.; Sagalowicz, Daniel; and Slocum, Jonathan (1978). "Developing a Natural Language Interface to Complex Data." In *ACM Transactions on Database Systems* 3(2), 105-147.

Hobbs, Jerry R. (1979). "Coherence and Coreference." In *Cognitive Science* 3(1), 67-90.

Reichman-Adar, Rachel (1984). "Extended Person-Machine Interface." In *Artificial Intelligence* 22, 157-218.

Sidner, Candace L. (1983). "Focusing in the Comprehension of Definite Anaphora." In *Computational Models of Discourse*, edited by M. Brady and R. Berwick, 267-330. MIT Press.

Craig, James A.; Berenzer, Susan C.; Carney, Homer C.; and Longyear, Christopher R. (1966). "DEACON: Direct English Access and Control." In *Fall Joint Conference of AFIPS* 29, 365-380. San Francisco, CA.

Trost, Harald; Buchberger, Ernst; Heinz, Wolfgang; Hörtnagl, Christian; and Matiasek, Johannes (1988). "Datenbank-DIALOG: A German language Interface for Relational database." In *Applied Artificial Intelligence* 1, 181-203. Hemisphere Publishing Corporation.

Turner Ray (1987). "A Theory of Properties." In *Journal of Symbolic Logic* 52(2), 445-472.

Winograd, Terry (1972). *Understanding Natural Language*. Academic Press, New York.

Woods, Willian A. (1973). "Progress in Natural Language Understanding: An Application to Lunar Geology." In *Proceedings of AFIPS National Computer Conference*, 441-450.