SUMMARIZING NATURAL LANGUAGE DATABASE RESPONSES

Jugal K. Kalita

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania

Marlene L. Jones

Department of Computer Science
University of Waterloo
Waterloo, Ontario, CANADA

Gordon I. McCalla

Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan, CANADA

In a human dialogue it is usually considered inappropriate if one conversant monopolizes the conversation. Similarly it can be inappropriate for a natural language database interface to respond with a lengthy list of data. A non-enumerative "summary" response is less verbose and often avoids misleading the user where an extensional response might.

In this paper we investigate the problem of generating such discourse-oriented concise responses. We present details of the design and implementation of a system that produces summary responses to queries of a relational data base. The system employs a set of heuristics that work in conjunction with a knowledge base to discover underlying regularities that form the basis of summary responses. The system is largely domain-independent, and hence can be ported relatively easily from one data base to another. It can handle a wide variety of situations requiring a summary response and can be readily extended. It also has a number of shortcomings which are discussed thoroughly and which form the basis for a number of suggested research directions.

1 INTRODUCTION

Research into the diverse and complex issues involved in developing smart natural language interfaces to database systems has been going on for over a decade. Pioneering front-end systems such as REL (Thompson and Thompson 1975), LUNAR (Woods, Kaplan, and Nash-Webber 1972), ROBOT (Harris 1977), PLANES (Waltz 1978), REQUEST (Plath 1976), TORUS (Mylopoulos et al. 1976), and RENDEZVOUS (Codd et al. 1978) experimented with, among other things, various parsing formalisms (e.g. semantic grammars, transformational grammars, and augmented transition networks); different knowledge representation schemes (e.g. using production systems or semantic networks); and the use of clarification dialogues in disambiguating a user's query.

Recent research has addressed various dialogue issues that arise in natural language database interactions. Researchers such as Davidson (1982), Grosz (1977), Sidner and Israel (1981), and Webber (1978) have tackled problems such as the resolution of anaphoric references, the tracking of the user's focus of attention, and the generation of cooperative responses. In particular, the CO-OP system (Kaplan 1982) analyzes the user's presumptions in order to generate appropriate explanations for answers that may otherwise mislead the user. Janas (1979) takes a similar approach to generate indirect answers instead of providing direct inappropriate ones. Mays (1982) has developed techniques to monitor

0362-613X/86/020107-124$03.00

changes in the data base and provide relevant information on these changes to the user. McCoy (1982) and McKeown (1982) attempt to provide answers to questions about the structure of the data base rather than extensional information as to its contents.

In this paper we investigate another dialogue issue: the generation of "summary" rather than extensional responses. Joshi, Kaplan, and Lee (1977) mention this as an interesting issue, but so far as we know the generation of summary responses has not been subsequently studied to any great extent. Summary responses are formulated in terms of general characteristics, shared by the extensional response set, that distinguish responses in that set from other information in the data base. It is often the case, as we argue later, that summary responses are not only more succinct than extensional responses, but are often more appropriate as well. In order to explore the issue of summary response generation, we have constructed a system that produces summary responses from a small relational data base of student records. The system has two components: a domain-independent set of heuristics and a domain-dependent set of frames that "customize" the heuristics for a specific data base and a particular class of users. Since the frames are relatively easy to specify for a given database context, the summary response techniques developed here may well be widely applicable; that is, the system is to a large extent portable.

The paper goes more fully into the nature of summary responses and their usefulness. It describes the response generation system in detail, including a presentation of the frames and heuristics used in the student database example. Extensive discussion is devoted to the strengths, weaknesses, and subtleties of this approach to summary response generation; and future research directions are outlined. A readable overview of the system can be found in Kalita, Colbourn (Jones) and McCalla (1984); a detailed presentation is available in Kalita (1984).

2 The Nature of Summary Responses

An important convention of human conversation is that no participant monopolize the discourse, ensuring that control can be shared (Joshi, Kaplan, and Lee 1977). For example it is often considered inappropriate for a speaker to respond with a lengthy list of data; a shorter non-enumerative response is, at times, more appealing. Lengthy response sets could be summarized, or defined by a characteristic or attribute. For instance, the question

Q1: Which employees engage in profit sharing?

may be answered by listing the extension of a set containing perhaps, a long list of names, or by the intensional response

S1: All vice-presidents.

Such summary answers avoid unnecessary and distracting details, and more important, they do not mislead the user.

As another example, consider the query Q2 given below (from Reiter et al. 1983):

Q2: Which department managers earn over $40k per year?

S2-1: Abel, Baker, Charles, Doug.

S2-2: All of them.

Response S2-1 is what might be expected of an existing system; response S2-2, the summary response, is normally more appropriate if conversational principles and practices are to be adhered to. By enumerating managers who *earn* over $40k, the first response implies that there are managers who *do not* earn that much. Such a **scalar implicature** follows from the cooperative principle in conversation (Grice 1975:45) that requires a speaker to make his/her "conversational contribution", such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which [he/she is] engaged". There are four maxims that derive from this principle:

(i) the maxim of *quantity* – make the contribution as informative as desired but not more so;
(ii) the maxim of *quality* – do not say what is believed to be false or that for which evidence is lacking;
(iii) the maxim of *relation* – be relevant;
(iv) the maxim of *manner* – avoid obscurity, avoid ambiguity, be brief, be orderly.

In the case of query Q2 above, the responder would only enumerate positive instances if he/she *could not* say the more informative *All of them.* Thus, S2-1 might mislead a user, who would expect the system to respond with S2-2 if it were true. Of course, the Gricean maxims must be viewed as being phrased relative to the responder's perceptions of the user's knowledge. Thus, the responder would have to know exactly what particular knowledge a given user had before being able to decide with certainty which responses are likely to be misleading. The usefulness of modelling the user will be discussed further in section 6.

In general, summary responses are aimed at meeting the maxims of quantity and manner. As Q2 illustrates, a summary response can be (somewhat paradoxically) more informative than an extensional response. It can also be briefer and less obscure than an extensional response. It is important to make only the relevant summary responses. Our system tries to maintain relevance through the use of a knowledge base tailored to meet the expectations of different classes of users (see section 4.2). Since we don't produce responses not satisfied by the data, the maxim of quality is not changed by the generation of summary responses. It *is* possible, however, to produce a summary response which itself violates Grice's maxims. For example, assume the query *Who passed CMPT 110?* were posed to a summary response generation system. Producing the answer *All students who got over 50%* (if this happened to be true in the current data base) would normally violate the maxims of quantity and relation in that most users would be given information they already knew. Similarly, producing the answer *All students who got over 60%* (if this happened to be true in the current data base) might mislead a user into thinking that 60% was a passing grade, hence violating the maxim of quality (actually, Joshi's generalization of this maxim since, strictly speaking, the answer is truthful). Some techniques are incorporated into our system to help reduce the chances of this kind of thing occurring (see section 4), but it is still a problem, as we discuss in the concluding section.

Generation of summary responses is analogous to a reversal of the interpretation process. A natural language question is interpreted into one or more propositions the data in the answer must satisfy, and then the appropriate data is retrieved. In a conventional database management system (DBMS), this extensional response is the only possible answer. But, we want to go back from the extensional data to predicates describing characteristics of the data and from there to natural language. Consider the query, *Which employees use a company car?*. The internal form into which this question is interpreted might be

(employee uses car) & (car belongs to company)

(The actual internal notation used in our system is more complicated than this – see section 4.3). A conventional DBMS would produce a response consisting of a set of employee names and possibly other relevant information about them. But, we want to obtain a descriptive answer, such as

(employee is president) $\vee$ (employee is vice-president)

which in turn can be expressed in natural language as

The president and the vice-presidents.

Hence, we must obtain a description that is true of the relevant data and present the description to the questioner instead of providing the actual data values that satisfy the propositions set forth in the question.

It is possible for a system to arrive at such concise responses from an extended database schema by employing a heuristic search of the extensional data for the existence of "interesting" patterns. In the next section we overview a system for producing summary responses.

3 Overview of the System

We have designed a system that produces summary responses to queries posed to a simple relational data base of student records. In order to concentrate on the pragmatics issues underlying the generation of summary responses, we ignore the complexities of starting with, and eventually producing, surface language. Instead, the system starts with predicates representing the user's query and produces predicates representing a summary response.

The flow of control in the system is simple. The user's query is formulated in an internal form which is understood by the underlying database management system. This internal form is discussed more fully in section 4.3. Using this query, the DBMS obtains the extensional response set, that is, the tuples that satisfy the user's query. After the data is accessed, the system consults its knowledge base to try to formulate a summary response. A prime component of this knowledge base is a set of heuristics used to find interesting non-enumerative patterns. As soon as a heuristic succeeds in discovering such a pattern, the system terminates the search and produces the response as dictated by the successful heuristic. This response is also in an internal notation identical in form to that used to represent the input. If all heuristics fail, the system reports its inability to produce a descriptive response. In any event, the user may ask the system to produce an extensional list of the data if desired.

Let's look at the knowledge base in slightly more detail. In order for the system to provide meaningful descriptive responses, the user's conceptions regarding the nature and contents of the data base must be taken into account. Without a separate knowledge base, this would be impossible. The knowledge base is employed to outline strategies for obtaining summary responses, to ensure that the qualitative responses generated are appropriate, and to produce salient information for describing the data that satisfy a query. The knowledge base consists of two distinct parts: the **heuristics**, and the **frames** for the relations and attributes.

The heuristics guide the search for "interesting" patterns in the data; the frames assist in determining "interestingness". The heuristics are the procedural part of the system's knowledge. There are several heuristics, including the equality, inequality, range, conjunction, disjunction, and foreign-key heuristics. They are ordered according to the complexity of the search procedures involved and are tried in this order so that the easiest (and usually the simplest to understand) summary response is found first.

The second part of the system's knowledge is represented by frames which encode useful information about the relations in the data base and their attributes. There are two types of frames: **relation frames,** which suggest ways of joining relations together in order to facilitate the discovery of elaborate patterns in the data; and **attribute frames,** which give characteristics of various attributes in the relations in order to aid the determination of relevant and interesting patterns.

Currently, both the frames and the heuristics must be prespecified by the system designer, rather than automatically created by the system to suit a given database context. However, this isn't a big problem since the heuristics are domain-independent and, hence, may be used with any other database domain without modification. And, although the frames must be tailored to reflect characteristics of the particular data base and user, the frame notation is sufficiently straightforward that it seems possible for a database manager to be able to do it relatively easily.

4 Details of the System

4.1 The Relation and Attribute Frames

The sample relational data base used in our implementation consists of three relations:

- STUDENTS,
- COURSE-DESCRIPTIONS, and
- COURSE-REGISTRATIONS.

The data base stores useful information about graduate students and the courses in which they register. The relations and their attributes are shown in Figure 1. Key attributes are shown in italics.

The current relation frames are very simple. Each frame corresponds to an actual relation in the data base; it provides the possible links with all other relations. In other words, these frames define all lossless joins of two relations. In cases where a direct join is not possible between two specific relations, the frame contains the name of a third relation that must be included in the join. If two relations R_1 and R_2 can be directly joined through attributes A_1 in R_1 and A_2 in R_2, the corresponding entry in the LINKS slot is

$$((R_1\ R_2)\ (A_1\ A_2)).$$

If the relations R_1 and R_2 cannot be joined directly, but can be indirectly joined through a relation R_3, the corresponding entry in the LINKS slot of the relation frames for R_1 and R_2 is

$$((R_1\ R_2\ R_3)\ (A_1\ A_{31})\ (A_{32}\ A_2)).$$

The first sublist indicates that the relations R_1 and R_2 can be indirectly joined through relation R_3. The second sublist indicates that R_1 and R_3 can be joined using the attribute A_1 in R_1 and the attribute A_{31} in R_3. Similarly, the relations R_3 and R_2 can then be joined through the attribute A_{32} in R_3 and A_2 in R_2.

For the STUDENTS relation under consideration, the relation frame can be seen in Figure 2. The relations STUDENTS and COURSE-REGISTRATIONS may be joined through the fields STUDENT-ID-NO in STUDENTS and STUDENT-ID in COURSE-REGISTRATIONS. The relations STUDENTS and COURSE-DESCRIPTIONS cannot be joined directly; the join has to be performed through the relation COURSE-REGISTRATIONS. STUDENTS and COURSE-REGISTRATIONS are linked through the fields named above. COURSE-REGISTRATIONS and COURSE-DESCRIPTIONS are joined through the COURSE-NO field in both these relations.

The information in the relation frames is employed when the system fails to produce a non-enumerative answer after exhausting all the heuristics that deal with only one relation. The system then attempts to find a descriptive expression considering another relation with which the original or target relation has some common join-attribute(s).

Relation frames allow the database manager the flexibility of naming attributes differently in different relations. They also can be used to restrict the types of joins that can be undertaken (i.e. not all possible joins need to be specified). Except for these distinctions, it would be relatively straightforward to generate the relation frames automatically.

STUDENTS:
    (*STUDENT-ID-NO*, NAME, NO-OF-YEARS-COMPLETED,
    NATIONALITY, NATURE-OF-FINANCIAL-AID, NO-OF-COURSES-THIS-TERM,
    NO-OF-COURSES-COMPLETED, TAKING-MAKEUP-COURSES?,
    CUMULATIVE-GPA, UG-MAJOR)

COURSE-DESCRIPTION:
    (*COURSE-NO*, COURSE-NAME, COURSE-LEVEL)

COURSE-REGISTRATIONS:
    (*COURSE-NO, STUDENT-ID*, OFFERED-IN-TERM, COURSE-GRADE)

Figure 1. Relations and Attributes in Graduate Student Data Base.

```
Relation-Name: STUDENTS
Links:
 (((STUDENTS COURSE-REGISTRATIONS)
   (STUDENT-ID-NO STUDENT-ID))
  ((STUDENTS  COURSE-DESCRIPTIONS COURSE-REGISTRATIONS)
   (STUDENT-ID-NO STUDENT-ID) (COURSE-NO COURSE-NO)))
```

Figure 2. Relation Frame for the Relation STUDENTS.

In addition to the relation frames, the system is provided with a number of attribute frames, each of which corresponds to an actual attribute in the data base. Attribute frames are critical in this approach to summary response generation and thus are described in some detail. Attribute frames allow important attributes and meaningful attribute values to be specified in advance. Together with the heuristics, they give our system many of the abilities of McCoy's (1982) ENHANCE system to reflect a user's preconceived notions as to which patterns of data are meaningful and which are not. A different set of attribute frames can be designed for each type of user (presumably by the database manager), thus allowing user modelling of a sort to be implemented.

Attribute frames guide the system in describing the data on the basis of attributes whose values serve to partition an entity class (represented by a relation in the data base) into two mutually exclusive subclasses, namely the part of the entity class that satisfies the user's query and the part that does not. As pointed out by Lee and Gerritsen (1978), some partitions of an entity class are more meaningful than others. Our system employs attribute frames to determine which attributes should be used for describing a partition and which resulting classifications are meaningful. Figure 3 shows an attribute frame for the attribute NATIONALITY in the STUDENTS relation.

```
Name:- (NATIONALITY, STUDENTS)
Nature-of-Attribute:- String of characters
Distinguishing-Values:-
      (((Canadian) (=) (≠ foreign))
       ((U.K. U.S.A. Australia ...)
             (member-of English-speaking-countries))
       ((U.K. France ...)
             (member-of Europe))
        ...)
Potential-range:- Any member from a given list of countries
Round-off-to-be-done?:- Not applicable
Preference-Category:- 1
```

Figure 3. Attribute Frame for NATIONALITY.

The NAME slot contains the internal name of the attribute, i.e. the name under which it is stored in the data base, and the name of the relation in which it occurs. If the attribute occurs in more than one relation, this field contains an entry for each relation. The general format of the contents of this slot is

(Attribute-Name-in-Relation-1 Relation-1-Name)
[(Attribute-Name-in-Relation-2 Relation-2-Name) ...]

The expression within the "[ ]" brackets is optional. The three dots indicate that an arbitrary number of repetitions of the immediately preceding expression is allowed. In the case of the attribute frame of Figure 3, the NAME slot indicates that the frame represents information about the NATIONALITY attribute in the STUDENTS relation.

The second slot, NATURE-OF-ATTRIBUTES, contains information regarding the type of values contained in the field – e.g., numeric, character, or boolean. The NATIONALITY attribute assumes character values.

The DISTINGUISHING-VALUE slot provides information for distinguishing a subclass of an entity from other subclasses. This slot stores any distinguishing values the attribute may take. These values are crucial in producing descriptive responses to the user's queries, so some time will be spent elaborating this idea. The slot contains one or more clauses, each of the following format:

```
(((list-of-attribute-values-1 (applicable-operator-1-1 [denomination-1-1])
                              [(applicable-operator-1-2 [denomination-1-2])]
                              ...)
```

If the actual values of the attribute satisfy "applicable-operator-1-1" with respect to the contents of the list "list-of-attribute-values-1", the actual values may be termed as "denomination-1-1" for producing responses. If the value of "denomination-1-1" is null, no special names can be attached to the actual values of the attribute.

Looking at the NATIONALITY attribute frame of Figure 3, a number of distinguishing values have been specified. Consider the clause ((Canadian) (=) (≠

foreign)). The value "Canadian" is a distinguishing value. The term "(=)" indicates that it is possible to identify a class of students using the descriptive expression "NATIONALITY = Canadian". If NATIONALITY ≠ "Canadian", the student may be referred to as a "FOREIGN" student. Similarly, if the value stored for a student under the attribute NATIONALITY is a **member** of the set (U.K. U.S.A. Australia ...), he/she may be designated as coming from an English-speaking country. Finally, if the student has value U.K., France, etc. for NATIONALITY, he/she may be considered to be from Europe.

Distinguishing values correspond to key values that naturally divide the values in a domain into distinct classes. In this sense they are very similar to McCoy's (1982) "very specific axioms", although how they interact with heuristics to produce summary responses is different. To illustrate, for most users the value 18 of an AGE attribute is a distinguishing value dividing children from adults; 65 is a distinguishing value separating adults from senior citizens. Other values are not important and, therefore, should not be considered to be "distinguishing". Similarly, suppose that a grade point average of six or greater is necessary for a graduate student to register in four courses rather than the usual three courses. The value "6", then, can be considered to be a distinguishing value for the CUMULATIVE-GPA attribute. This would allow questions like *Which students are taking four or more courses?* to be answered with *All students with GPA of six or higher* rather than with the response *All students with GPA of 6.52 or higher* which might be true of the current data. The latter response is inappropriate because it violates the maxim of quality in that it might mislead the user into thinking that 6.52 is a significant value in the University. (See Q8-S8 in section 4.2.4 for the details as to how the proper summary response for this kind of question is generated by our system.)

Returning to the NATIONALITY frame of Figure 3, the distinguishing values specified there would make it possible for our system to answer the question *Which students are taking the "Intensive English" course in the Fall term?* with the response *Most entering foreign students from non-English speaking countries* rather than the misleading answer *All students from China, Iran, and France,* which might happen to be true currently. Once again, the latter response violates the maxim of quality, a common occurrence if summary responses are not carefully tuned to reflect significant domain subdivisions.

The DISTINGUISHING VALUE slot enables the database manager to specify classifications that he/she would a priori like to appear meaningful to the user in descriptive responses. Without this information the system may fail to faithfully reflect the user's perceived notions regarding appropriate partitioning of entity classes. By changing the distinguishing values, the database manager can adapt the system to serve the needs of a variety of users. Although it isn't our concern here, it would even be possible to remove all distinguishing values and hence have the system produce no summary responses. For any given class of users, the database manager will need to specify all of these distinguishing values by hand, but once they are specified, they can be used by many different heuristics in many different situations for as long as the database structure remains the same, even if the tuples in the data base change. Further examples of the use of distinguishing values and how they interact with the heuristics will be presented shortly.

Let us return at last to the other slots in an attribute frame. The POTENTIAL-RANGE slot provides an approximate range in which the values of the attribute may lie. The information in this slot is employed in conjunction with the range heuristics which are discussed in the next section. In the NATIONALITY attribute frame of Figure 3, the potential range would be specified in terms of a long list (not shown) of possible countries of origin.

It is sometimes necessary to round off values of numeric attributes in order to produce answers with acceptable range specifications. However, not all numeric attributes can be rounded. Whether rounding is allowable for a particular attribute depends on several factors including the type of values the attribute can assume (i.e., integer, real, etc.) and the potential range of its values as well as other attribute characteristics. The ROUNDING-TO-BE-DONE? slot contains a boolean value indicating whether rounding is appropriate for the particular attribute under consideration. It obviously is not for the character values of the NATIONALITY attribute frame. Straightforward as it may seem, rounding allows our system to avoid violating Grice's maxims of manner, specifically by making answers less obscure.

It is often more useful to provide descriptive answers on the basis of certain preferred attributes. For example, in the STUDENTS relation, it is more "meaningful" to provide answers on the basis of the attribute NATIONALITY or UG-MAJOR rather than STUDENT-ID-NO or AMOUNT-OF-FINANCIAL-AID. However, it is impossible to give a concrete weight regarding each attribute's preferability. Therefore, we have classified the attributes into several groups; all attributes in a group are considered equally useful in producing meaningful qualitative answers to queries. The groups for the STUDENTS relation are given in Figure 4.

This classification means that it is preferable and more useful to produce descriptive responses using the attributes in group 1 than the attributes in group 2, and the attributes in 2 are preferable to 3, which are in turn preferable to 4. This categorization is done by the database manager, based on his/her judgement as to the perspectives of the various classes of users. In the slot PREFERENCE-CATEGORY, there is an entry corresponding to each relation the attribute occurs in. The information in this slot ensures that the system chooses a description based on the most salient attribute for producing a

```
1: (NATIONALITY, CUMULATIVE-GPA, UG-MAJOR)
2: (NO-OF-YEARS-COMPLETED, NATURE-OF-FINANCIAL-AID)
3: (NO-OF-COURSES-THIS-TERM, NO-OF-COURSES-SO-FAR, TAKING-MAKEUP-COURSES?)
4: (STUDENT-ID-NO, NAME)
```

Figure 4. Preference Categories for the Relation STUDENTS.

```
Name:- (CUMULATIVE-GPA STUDENTS)Nature-of-Attribute:- Real
Distinguishing-Values:-
   (((0.00 2.00) (in-the-range-of poor))
    ((2.00 4.00) (in-the-range-of satisfactory))
    ((4.00 6.00) (in-the-range-of good))
    ((6.00 7.00) (in-the-range-of excellent))
    ((7.00 8.00) (in-the-range-of outstanding))
    ((2.00) (≥) (≤))
    ((6.00) (≥) (≤)))
Potential range:- (0.00-8.00)
Rounding-to-be-done?:- Yes
Preference-Category:- 1
```

Figure 5. Attribute Frame for CUMULATIVE-GPA.

```
Name:- (NO-IF-COURSES-THIS-TERM STUDENTS)
Nature-of-Attribute:- Integer
Distinguishing-Values:- (((2) (≤ light-load))
                         ((3 4) (member-of normal-load))
                         ((5) (≥ heavy-load)))
Potential-Range:- (0-6)
Rounding-to-be-done:- No
Preference-Category:-3
```

Figure 6. Attribute Frame for NO-OF-COURSES-THIS-TERM.

response. The preference category of the NATIONALITY attribute of Figure 3 is 1.

Preferred attributes perform for our system the same function that McCoy's (1982) "important attributes list" does for the ENHANCE system. We go further than McCoy in specifying several preference categories, rather than having one long list. Although all attributes are assigned a preference category in this example, we can, like McCoy, leave out unimportant attributes altogether if it is appropriate to do so.

Let us now look at two more attribute frames. Figure 5 shows the frame for the attribute CUMULATIVE-GPA in the STUDENTS relation. From this, it is clear that CUMULATIVE-GPA takes real values in the range 0.00 to 8.00. If CUMULATIVE-GPA is in the range 2.00-4.00, it may be termed "poor."; Similarly, if it is in the range 4.00 to 6.00, it is considered as "good", and so on. If none of the first five clauses in the DISTINGUISHING VALUE slot is satisfied, the system attempts to use the last two clauses. The clause ((2.00) (≥) (≤)) says that we can use expressions such as "GPA ≥ 2.00" or "GPA ≤ 2.00", which cover a wider range than the first five clauses (e.g. *Which students are allowed to continue?* might be answered *All students with GPA of 2 or more.* – see also Q7-S7 in section 4.2.4). It should be noted that these expressions may be used only if all values for the attribute GPA in the selected tuples satisfy the corresponding condition. However, we cannot use expressions of the form "GPA = 2.00". We avoid using equalities for attributes that assume rational values. The clause ((6.00) (≥) (≤)) conveys a similar idea.

Figure 6 shows the frame for the attribute NO-OF-COURSES-THIS-TERM in the STUDENTS relation. From this figure, one can conclude that the attribute NO-OF-COURSES-THIS-TERM assumes integer values in the range 0-6. If this field has a value ≤2, it may be termed "light-load". If NO-OF-COURSES-THIS-TERM is either 3 or 4, it is "normal-load". If the value of the attribute is ≥ 5, it is "heavy load". The values of the thresholds shown here are applicable in the case of graduate students. These values would, obviously, be different if we considered a data base of undergraduate students.

Currently, the attribute frames are static entities with their contents being defined a priori by the database manager to reflect the expectations of one set of users. Of course it is possible to have many different sets of attribute frames for many different classes of users, but a better approach might be to allow the user to alter the contents of these frames interactively to suit his/her own idiosyncratic perceptions of the information in the data base. This would require us to figure out how to present the frames and the possible changes to the user, something we haven't done as yet. Even more difficult would be the automatic creation (and later adjustment) of the attribute frames as the result of feedback from a particular user or class of users. This is clearly a major research issue beyond the scope of our current concerns.

4.2 THE HEURISTICS

As mentioned earlier, the heuristics employed in the system are procedural in nature. In conjunction with the frames described above, they guide the system to search for various interesting patterns that distinguish the tuples describing the query response from the rest of the tuples in the data base. The interesting patterns are similar to McCoy's (1982) "distinguishing descriptive attributes", although we use them to produce summary responses rather than to answer questions on database structure. Our use of a number of specially designed heuristics using frames to produce meaningful responses is also different from McCoy's approach, which uses three different kinds of axioms to control a general search procedure.

In order to help overcome possible problems of combinatorial explosion (mentioned as a problem for McCoy's ENHANCE as well), the heuristics are linearly ordered according to the complexity of the required search procedures. Hence, the system first searches for simple patterns; the complexity of the response patterns grows as later heuristics are employed. This ordering of the heuristics assumes that, if more than one descriptive answer can be obtained for a query, it is sensible to produce the "simplest" one. It would be easy to change this if more sophisticated termination conditions for the search were desired.

We assume that the natural language query has been parsed and transformed to an internal form, and the required data have been accessed. The heuristics are applicable only after the tuples that satisfy the user's query are at hand. Let T_{qual} be the set of tuples that satisfy the user's query, and T_{unqual} be the rest of the tuples in the relation relevant to the current query.

4.2.1 THE EQUALITY HEURISTIC

The **equality heuristic** is the most elementary of all the heuristics. It corresponds to the usage of everyday words such as *all, everybody* and *everyone*. To start our discussion, we present a formal specification of the heuristic.

> Determine if all data values appearing as the value of a particular attribute A in T_{qual} are the same (say, α). α must be a DISTINGUISHING VALUE in the domain of values for attribute A. If so, and if no tuple in T_{unqual} has the value α for the attribute A, the general formulation of the response is:
>
> *All tuples having the value α for attribute A.*

An example is the question-answer pair Q3-S3:

Q3: Who are the Canadian students with GPA of 7.5 or higher?
S3: All students receiving NSERC scholarships.

For applying this heuristic, the value α of the attribute A must have some "distinguishing" importance in the domain. In the above example, the attribute under consideration is NATURE-OF-FINANCIAL-AID. The value NSERC is considered to be a DISTINGUISHING VALUE in the domain of values that the attribute NATURE-OF-FINANCIAL-AID can take.

The equality heuristic may also be applied to certain numeric attributes. Consider the following question and answer pertaining to the graduate student data base.

Q4: Which students have completed less than 5 courses?
S4: All first year students.

Here, the value of the attribute NO-OF-YEARS-COMPLETED is 0 for all tuples that satisfy the query Q4. Also, among the unqualified tuples, there is none in which NO-OF-YEARS-COMPLETED = 0. Finally, the value 0 distinguishes first year students from others, according to the attribute frame for NO-OF-YEARS-COMPLETED.

Before leaving the equality heuristic, it should be noted that Q1-S1 (*Which employees engage in profit sharing? – All vice-presidents.*) from section 2 could be handled by the equality heuristic (all employees engaging in profit sharing have the rank "vice-president"; nobody who isn't engaging in profit sharing has this rank).

4.2.2 THE INEQUALITY HEURISTIC

The dual of the equality heuristic is the **inequality heuristic**; instead of looking for equalities, the system searches for inequalities. Formally, the heuristic may be stated as,

> Determine if each data value for a particular attribute in T_{qual} is not equal to some particular value γ and all tuples in T_{unqual} have that value. This value γ must be a DISTINGUISHING VALUE in the domain of the values for attribute A. The general formulation of the response is
>
> *All tuples with value of attribute A $\neq \gamma$.*
>
> In order to produce the required response, the system must make certain that A $\neq \gamma$ is not true in any of the tuples which do not satisfy the user's query.

Let us consider an example. In the student data base, the value "Computer Science" for the attribute UG-MAJOR may be considered a distinguishing value. This allows us to produce a response such as

All students with majors other than Computer Science.

or, equivalently,

All non-Computer Science majors.

as in the following question and answer pair:

Q5: Which students have taken more than six courses?

S5: All students with non-Computer Science undergraduate background.

At the same time, we may avoid producing a response such as (say)

All students from departments other than Mechanical Engineering,

if *Mechanical Engineering* is not of interest to us. Thus, it is clear that the specification of distinguishing attribute values is dependent on the user's conception of the data as well as the application under consideration. It should be noted that phraseology subtleties such as the differences between *All non-Computer Science majors*, *All students with majors other than Computer Science*, or *All students with non-Computer Science undergraduate background* are not reflected in different internal notations, but are the responsibility of the natural language generation component which we haven't developed as yet. Such subtleties can be quite important, but are left for future research. The whole issue of natural language generation (and interpretation) is discussed further in section 4.3.

4.2.3 MODIFICATION OF THE EQUALITY AND INEQUALITY HEURISTICS

If the equality or inequality heuristics are not applicable in their pure form and there are a "few" ("few" depends on the relative number of tuples in T_{qual} and T_{unqual} and some other factors) tuples in T_{unqual} that do not satisfy the requirement of the heuristic, a modification of the response produced by the heuristic may be presented to the user. An example of such a modification is seen in the following:

Q6: Which students are receiving University scholarships?

S6: All but one foreign student. In addition, two Canadian students are also receiving University scholarships.

4.2.4 RANGE HEURISTICS

These heuristics determine if the data values for an attribute in the tuples in T_{qual} are within a particular well-defined range. There are two main types of range heuristics – one is concerned with maximum values and the other with minimum values. The first of these, the **maximum heuristic**, may be formally stated as,

> Determine if all data values for attribute C in T_{qual} are below some maximum (say, β), and there is no tuple in T_{unqual} with values for $C \leq \beta$. This value β must have some "distinguishing importance" in the domain of the values of attribute C. In this case, the general formulation of the response is
>
> *All tuples with the value of attribute* $C \leq \beta$.

An illustrative example is

Q7: Which students have been advised to discontinue studies at the University?

S7: All students with a cumulative GPA of 2.0 or less.

Here GPA = 2.0 is assumed to have some "distinguishing importance" in the field of numbers representing GPAs of students (i.e., a value that may be "generally" used to partition the set of all possible GPAs into two classes: ones above 2.0, and ones equal to or lower than 2.0). The maximum heuristic is generally applicable in the case of numeric attributes.

Similarly, the **minimum heuristic** may be formally specified as,

> Determine if all data values for attribute C in T_{qual} are above a certain minimum (say, δ) and there are no tuples in T_{unqual} with value for $C \geq \delta$ • δ must have some "distinguishing importance" in the domain of the values of attribute C. The general formulation of the response is
>
> *All tuples having the value in column* $C \geq \delta$.

An illustrative example is

Q8: Which students are taking four or more courses?

S8: All students with GPA of six or higher.

When the tuples in T_{qual} satisfy both the maximum and the minimum heuristics for the same attribute A, we get a **range specification.** Let α be the minimum value and β be the maximum value of the attribute A in T_{qual}. Then the response can be modified as

All tuples with value of attribute ranging from α *through* β.

An example of an answer with range specification is

Q9: Who are the students taking courses in second year?

S9: All students who have completed between 3 and 5 courses so far.

There are several rules that should be followed while producing answers in terms of ranges. Some of the rules employed in the current implementation are given below. These rules are fairly arbitrary, but rules like them will be necessary to prevent summary responses from themselves violating Grice's maxims, especially the maxims of manner and quality.

- If the upper limit of the actual range for an attribute is the maximum potential value for the attribute, it is better to modify the answer as *more than* α where α is the lower limit of the actual range. For example, if for an attribute A the upper limit of the maximum potential range is 1000, instead of providing a response *between 750 and 1000*, it is advisable to say *more than 750* if Grice's maxim of manner (be brief) is to be satisfied.
- A similar action is taken at the other end of the scale. For example, if the lower limit of the maximum potential range is 0, instead of responding as *between 0 and 200*, we might answer as *less than 200*.
- The actual range specified in an answer should not be more than 75% of the potential range of the attribute values. The particular choice of 75% is not sacrosanct, but the rule itself is important if we are to avoid the problem of producing a response that essentially covers

the entire range of potential attribute values. Such a response would mislead the user into thinking that there existed values outside of this range, which would violate Grice's maxim of quality.

- The actual range specified in an answer should not be so small as to identify the actual tuples that constitute the answer. For example, we should not produce a response such as, *All students with student-id-no between 821661 and 821663.* In fact, such answers are not brief when compared with the size of the set of tuples they qualify. Moreover, they can mislead the user into thinking that there are many more tuples than there actually are in the response set.

These violations of the maxims of manner and quality should be avoided.

While producing range specifications, it is often necessary to round off the upper and lower limits in case of numeric attributes. For example, instead of saying *Students with GPA between 6.06 and 6.92* we may as well say *Students with GPA between 6.00 and 7.00.*

Rounding cannot be done for all numeric attributes. The applicability of the rounding operation depends on several factors including the nature of the values the attribute takes – e.g. whether they are an integer or rational, and their potential range.

- In case of integer values, if the potential range is "small", rounding should be avoided. For example, the field NO-OF-YEARS-COMPLETED in a student data base has a tight potential range (0-5 years). In this case, if we have data values between 2 and 4 years, we should not round off and say *between 2 and 5 years.*
- For integer values, if the potential range is wide, rounding off may be done (except for some cases discussed below). For example, the expression *Students with marks between 61 and 78* may be rounded to *Students with marks between 60 and 80.* However, for this rounding to be correct, it is necessary to ensure that there are no tuples in T_{unqual} with marks 60, 79 or 80.
- There are certain attributes that are integral and do not allow approximation by their inherent nature. One example of such an attribute is STUDENT-ID-NO. A student identification number 82116 cannot be approximated as STUDENT-ID-NO = 82115 or STUDENT-ID-NO = 82120. Similarly, we cannot round the attribute YEAR-OF-BIRTH in many circumstances. This decision whether rounding should be done or not is often subjective. Hence, this information must be provided by the system builder and stored in the knowledge base.
- If an attribute assumes non-integer (i.e., rational) values, the system may nearly always proceed with rounding. It may be possible to find counter examples to this assertion in some database domains. However, for the purpose of the current implementation, we accept this assumption to be true at all times.

It should be noted that the heuristics explained above are applicable when a single attribute of the relevant relation is considered. If no such heuristic can be successfully applied to the pertinent data, the system attempts to use one of the conjunction or disjunction heuristics jointly on two or more attributes.

4.2.5 CONJUNCTION HEURISTIC

The conjunction heuristic is the first of the complex heuristics involving more than one predicate. Usually, each of these predicates involves a distinct attribute in the data base, although it is possible that two or more predicates relate to values of the same attribute. These heuristics provide the system with the facility to use common connectives such as *and* and *or.*

The conjunction heuristic is expressed succinctly in the following paragraph.

> If all values of an attribute C in T_{qual} satisfy a relation R (in the mathematical sense), and there are tuples in T_{unqual} that also satisfy the same relation R, determine via the above heuristics if there is/are some "interesting" distinguishing characteristic(s) that the set T_{qual} satisfies, but the set of tuples in T_{unqual} satisfying the relation R do not. Let us call the distinguishing characteristic(s) D. The general formulation of the response is
>
> *All tuples that satisfy the relation R and have the characteristics D.*

An example is,

Q10: Which students are working as T.A. or R.A.?

S10: Students who have completed more than 1 year at the University and who are not employed outside the University.

All the tuples in T_{qual} resulting from Q10 are found by the system to have the values for the attribute NO-OF-YEARS-COMPLETED > 1. However, the system finds that there are some tuples in T_{unqual} that also have values greater than 1 for the attribute NO-OF-YEARS-COMPLETED. Let us call these tuples T_{equal}. Next the system attempts to find some characteristics that distinguish T_{equal} from T_{qual}. It finds that in T_{equal} the field NATURE-OF-FINANCIAL-AID = OUTSIDE-JOB for all tuples whereas in T_{qual}, NATURE-OF-FINANCIAL-AID = UNIVERSITY-SCHOLARSHIP for all tuples. After finding this difference, the system is able to qualify the phrase *Students who have completed more than 1 year in the University* by the phrase *who are not employed outside the University* to produce the complete response.

4.2.6 DISJUNCTION HEURISTIC

If none of the above heuristics can be applied successfully, the system attempts to use the **disjunction heuristic.** As is evident from the nomenclature, this heuristic enables the system to formulate complex responses using the connective OR. Formally, this heuristic may be expressed as follows.

Divide the tuples in T_{qual} into a number of subsets and try to apply one of the heuristics explained earlier to each subset. If successful, the resulting response consists of several predicates connected by the relational operator OR ($\vee$). It has the generalized format

Tuples with (attribute$_1$ R_1 α_1) $\vee$ (attribute$_2$ R_2 α_2) [$\vee$ (attribute$_3$ R_3 α_3) ...]

where R_i's are relations (in the mathematical sense); α_i's are distinguishing values of the corresponding attribute$_i$'s.

While formulating responses with the disjunction heuristic, the number of such subsets should be restricted to two or three, if possible. If too many subsets are identified, it is difficult for the user to grasp all of them. If more than three subsets are presented, this approach is no more elegant than listing the data, which we are trying to avoid. The number of allowable subsets also depends on the number *n* of tuples in T_{qual}. If *n* is "large", the number of subsets one would consider acceptable may be somewhat higher.

It should be mentioned that in the generalized expression for the response, the various attribute$_i$'s may be the same attribute, or they may be different. In certain cases, the same attribute may partition the relevant information into two or more groups in distinct ways. An example showing three partitions based on the values of three different attributes is,

Q11: Which students are not receiving University scholarships?

S11: Students who are receiving NSERC scholarships or have cumulative GPA less than 6.0 or have completed at least two years at the University.

In attempting to answer Q11, the system finds that it is not possible to obtain an appropriate answer using the previous heuristics. It then checks to see if the tuples in T_{qual} can be divided into two or three separately identifiable subsets. In this case, it successfully partitions T_{qual} into three subsets – T_{qual-1}, T_{qual-2} and T_{qual-3} where

- T_{qual-1} consists of all tuples in T_{qual} for which NATURE-OF-FINANCIAL-AID = NSERC-SCHOLARSHIP,
- T_{qual-2} is the subset of all tuples in T_{qual} with CUMULATIVE-GPA < 6.00, and
- T_{qual-3} is the subset of tuples in T_{qual} for which NO-OF-YEARS-COMPLETED $\geq$ 2.

While subdividing the total response set T_{qual} into subsets, the system should ensure that no tuple in T_{unqual} satisfies the various disjunctive predicates.

4.2.7 FOREIGN-KEY HEURISTIC

If nothing satisfactory can be found employing all of the above heuristics, the system attempts to search other "related" relations to obtain a suitable response. A related relation is one with which the relation under consideration has some common or join attribute(s). Formally, the foreign-key heuristic may be stated as,

Obtain the tuples in the target relation R_t that satisfy the user's query. Let these tuples constitute a new relation R_n. Determine if the target relation R_t may be joined directly or indirectly with some other relation(s) in the data base by consulting the relation frame for R_t. Let these other relations be designated $\{R_j\}$ where maximum(j) = number of such "related" relations. Take join of R_n with the R_j's one at a time (these joins may be direct or indirect and are performed via the attributes specified in the relation frame). Project the resulting relation on the attributes of R_j and try to apply one of the previous heuristics to this resultant relation. Stop only when there is successful application of a heuristic for some R_j, or each relation R_j has been tried unsuccessfully.

As an example, consider the following question and the response to it:

Q12: Which students are taking CMPT 994?

S12: All students who have completed at least one year of studies.

While attempting to answer Q12, the system finds that the question pertains to the relation COURSE-REGISTRATION. However, it fails to obtain any interesting descriptive pattern about the tuples in T_{qual} by considering this relation alone. Hence, the system consults the LINKS slot in the relation frame for COURSE-REGISTRATION and finds that COURSE-REGISTRATION may be joined with the relation STUDENT via the fields STUDENT-ID-NO in STUDENTS and STUDENT-ID in COURSE-REGISTRATION. It takes a join of all the tuples constituting T_{qual} with the relation STUDENTS and projects the resulting relation on the attributes of the relation STUDENTS. Let us call these tuples $T_{new-qual}$. Next, it attempts to discover the existence of some pattern in the $T_{new-qual}$ tuples. Ultimately, it succeeds in producing the response given in S12 by employing a minimum range heuristic.

4.3 THE INTERNAL FORM OF A QUERY

The internal form of a query is

(Command Database-Identification Predicate-Form)

- **Command** is some operation to be performed, at the moment limited to the command OBTAIN, meaning obtain information from the data base.
- **Database-Identification** names a particular data base on which the command is to be carried out; in the current implementation, GRAD-STUDENT-RECORDS.
- **Predicate-Form** breaks down into
  - **(Predicate (Relation-Name Attribute-Name) Attribute-Value)**
    (e.g., (GREATER-OR-EQUAL (STUDENTS CUMULATIVE-GPA) 6))

  or
  - **(Conjunction Predicate-Form$_1$ Predicate-Form$_2$ ... Predicate-Form$_n$)**
    (e.g., (OR-ANY-OF

```
(EQUAL (STUDENTS
NATURE-OF-FINANCIAL-AID) NSERC)
(LESS-THAN (STUDENTS CUMULATIVE-GPA) 6)
(GREATER-OR-EQUAL (STUDENTS
NO-OF-YEARS-COMPLETED) 2) ) )
```

Common predicates such as EQUAL, NOT-EQUAL, LESS-THAN, and GREATER-OR-EQUAL, and conjunctions such as AND-ALL-OF, AND-ANY-OF, etc. can be handled. The internal form of the system output is defined in a similar manner except there is no command.

The following examples of queries and answers (taken from section 4.2) represented in this internal form should clarify this notation. First we look at a relatively simple case:

Q4: Which students have completed less than 5 courses?

```
(OBTAIN GRAD-STUDENT-RECORDS
(LESS-THAN (STUDENTS NO-OF-COURSES-COM-
PLETED) 5))
```

S4: All first year students.

```
(GRAD-STUDENT-RECORDS
(EQUAL (STUDENTS NO-OF-YEARS-COMPLET-
ED) 0))
```

Next we take a more complicated question-answer pair:

Q10: Which students are working as T.A. or R.A.?

```
(OBTAIN GRAD-STUDENT-RECORDS
(AND-ANY-OF
   (EQUAL (STUDENTS NATURE-OF-FINAN-
   CIAL-AID) TA)
   (EQUAL (STUDENTS NATURE-OF-FINAN-
   CIAL-AID) RA)))
```

S10: Students who have completed more than one year at the University and who are not employed outside the University.

```
(GRAD-STUDENT-RECORDS
(AND-ALL-OF
   (GREATER-THAN (STUDENTS NO-OF-YEARS-
   COMPLETED) 1)
   (NOT-EQUAL (STUDENTS NATURE-OF-FINAN-
   CIAL-AID) OUTSIDE-JOB)))
```

We do not want to downplay the difficulties of interpreting natural language into an internal form such as this, nor do we want to trivialize the difficulty of producing surface language responses from the internal form. However, parsing and natural language generation were not the central concerns of this research; we instead wanted to concentrate on the pragmatic issues underlying summary response generation in a natural language database interface. There is a plethora of work, of course, describing various approaches to parsing we could draw on should we want to extend our system. Possibly the most appropriate parsing strategy for this domain would be a keyword approach (e.g., Small 1980) where the input query is scanned for words indicative of attribute names or predicates relevant to the particular data base being queried. This approach might work well here because the target internal form is phrased only in terms of these domain specific attributes and predicates.

Similarly, generation could be in terms of catch phrases triggered by the presence of predicates or attributes in the internal form of the output. There is relatively less work on natural language generation on which to base a more sophisticated natural language generation component, but work such as McDonald's (1983) MUMBLE system might be usefully adapted to the determination of appropriate surface phraseology of summary responses. The approach taken in McKeown's (1982) TEXT system is also appealing in this regard since its area of application is data bases (albeit describing database structure rather than database contents). To adapt methods from either of these systems (or in fact from most other approaches to generation) would require a considerable enhancement of the knowledge base of our system, something that is currently beyond the scope of the research.

5 Implemenation Considerations

A system incorporating the details discussed above has been implemented in Franz Lisp on a VAX-11/750 running under UNIX[1] and has been tested on a data base of student records. The system was tested on a variety of questions. These included all of the examples Q3-Q12 where the system produced internal versions of the summary responses S3-S12. Further details of these examples (and others) are contained in Kalita (1984). The data base is currently very small (containing the records of only 25 students or so), so the average response time of the system was in the order of seconds, even for the most time consuming heuristics. A more meaningful analysis is a complexity analysis of the response time in terms of the number of tuples in the data base. With this in mind, in this section we examine implementation aspects of the system, including a complexity analysis of the various heuristics.

The system has two main components – one for data manipulation, the other to produce the summary responses. The data manipulation component enables the system builder to introduce new relations, new attributes, and new tuples into the relations. As new tuples are entered, various checks regarding the nature of the attribute values and the number of attributes are performed. The data manipulation component also accesses the data that satisfy a query and performs standard relational functions such as selection, projection, and lossless join. This component does not possess the sophistication of a standard database package. However, it is sufficient for the purposes of this research since the internal form of a query can be directly handled by the data manipulation routines.

The other main component of the system produces summary responses to a user's queries. First the user's input is read and checked for syntactic accuracy (i.e., that it follows the proper internal form, that it contains only references to valid names of relations and attributes, etc.). The query is then passed to the data manipulation component for data access. Returned are the two sets of tuples: T_{qual}, those tuples satisfying the user's query, and T_{unqual}, those that do not. The summary response component then regains control and invokes routines corresponding to the individual heuristics. The invocation of the heuristics is done successively in predetermined order until one of them is successful. There is some dependence among the heuristic routines since certain information, once obtained, can be shared. The heuristics receive assistance from the frames during the process of obtaining summary responses. These frames are stored as property lists associated with the relation and attribute names.

The heuristics attempt to determine a descriptive response by searching through T_{qual} and T_{unqual}. In the current implementation, the tuples are examined serially. Once a tuple has been accessed, various attribute values in it are tested in parallel to determine if they satisfy the requirement of a heuristic. Such tuple-serial attribute-parallel inspection of attribute values may increase processing time in some cases. However, on average, the time required for obtaining a response is considerably reduced since the tuples need not be accessed repeatedly for each candidate attribute under consideration.

While applying the equality heuristic for each attribute in the target relation, the system keeps a frequency count of different values that occur in the various attributes in T_{qual}. If at any time during the equality processing of T_{qual} the system finds more than three different data values for a particular attribute, it ignores the attribute during subsequent processing for the equality heuristic. If for a particular attribute, all values in T_{qual} are the same and this particular value is (a) a distinguishing value and (b) does not occur in any of the tuples in T_{unqual}, the system produces a response using the equality heuristic. If there are up to three different values that occur for an attribute in T_{qual} and do not occur in T_{unqual}, the system compares the dominant frequency with the other frequencies. In the current implementation, the system produces an answer using a modification of the equality heuristic if the other frequencies are less than 10% of the dominant frequency.

For the application of the inequality heuristic, the roles played by T_{qual} and T_{unqual} are interchanged. Otherwise, the processing is essentially the same.

For the range heuristics, the maximum and minimum values for each attribute in T_{qual} are found in tuple-serial attribute-parallel mode. If both heuristics are successful for a particular attribute, a response in terms of range specification is generated. The rules discussed in section 4.2.4 are applied for obtaining responses using the heuristics.

The disjunction heuristic is attempted when it is possible to divide the tuples in T_{qual} into two or three subgroups based on equality, inequality, or range heuristics. While applying the earlier heuristics, the system has retained information that may help in the application of the disjunction heuristic. However, application of the disjunction heuristic may necessitate a substantial amount of repetitive grouping and regrouping of tuples and may be expensive in its time requirements. Even so, success is not guaranteed.

The conjunction heuristic is successful when there are tuples in T_{unqual} that satisfy the predicate(s) satisfied by the tuples in T_{qual}. Let the tuples in T_{unqual} that satisfy the predicate(s) be called T'_{unqual}. To obtain an answer, the equality, inequality, and range heuristics are employed using T_{qual} and T'_{unqual} (instead of the usual T_{qual} and T_{unqual}) to find some distinguishing characteristics between the two sets. This distinguishing description is then used as a qualifier to obtain the final complete response.

The foreign-key heuristic involves a join and a projection, and finally the application of all previous heuristics. If the target relation has common join attributes with several other relations, joins may have to be performed with each such relation, and the process repeated again for each resultant relation.

If one of the heuristics succeeds, a response is generated in the format described above. If none of the heuristics succeeds, the extensional response T_{qual} is produced. The user can also ask for T_{qual} to be produced if he or she is unsatisfied with just the summary response.

In order to determine the implications of our approach to summary response generation, it is important to look at the computational complexity of the algorithms. The application of the equality, inequality, and the range heuristics takes time of the order of $O(N_a N_t)$ where N_a is the number of attributes in the target relation, and N_t is the number of tuples in the target relation (i.e., the sum of the number of tuples in T_{qual} and T_{unqual} for the target relation). Performance is improved if the value comparisons are done in parallel for all attributes in a tuple. This performance improvement results since the tuples need not be accessed for each attribute separately. However, this does not reduce the basic complexity involved in the determination.

The complexity of applying the disjunction heuristic is dependent on the nature of data distribution in T_{qual}. Successful application may involve a large number of permutations of the tuples for repetitive grouping and regrouping. This is the heuristic most likely to lead to a combinatorial explosion.

The conjunction heuristic takes time of the order $O(N_a N_t + N_a N_{t1})$ where N_{t1} is the sum of the number of tuples in T_{qual} and the number of tuples in T_{unqual} that satisfy some mathematical relation(s) satisfied by the tuples in T_{unqual} (i.e., T'_{unqual} defined earlier). This complexity can be arrived at only if we assume that the disjunction heuristic is not applied to determine the distinguishing characteristics between T_{qual} and T'_{unqual}. Otherwise, the time required will be $O(N_a N_t + N_a N_{t1}) + O_{dh}(T_{qual}, T'_{unqual})$ where O_{dh} is the time requirement for the application of the disjunction heuristic.

The foreign-key heuristic requires additional time for performing joins and projections. The number of joins that may be performed is a function of the number of relations with which the target relation has common join (direct or indirect) attributes. Indirect joins involve one or more additional simple joins. The complexity of the computations necessary after completion of the join and subsequent projection is the same as discussed in the preceding paragraphs.

The implications of these complexity bounds for large data bases cannot be ignored. Processing time for the simple equality, inequality, and range heuristics is linear in the number of tuples in the data base. This is about as good as can be expected, although it still may be quite slow if real time response is needed. Processing times for the disjunction, conjunction, and foreign-key heuristics can be substantially worse as juggling, rejuggling, and joining take place. If each query must be independently processed, we don't see much hope of improving on these times. However, it may be possible to add a "memory"to the system's knowledge base to keep track of previous responses and hence avoid re-accessing the data base for each query. The nature of such a memory and some of the implications are discussed in the next section.

6 Conclusions and Future Research

Generation of descriptive summary responses has important implications if interactions with a data base are to have the properties and constraints normally associated with human dialogue. Without these constraints, interactions with a DBMS can simply be viewed as dull factual exchanges between a human being and a machine. No doubt, the necessary data is obtained by the user, but these interactions lack the "intelligence" and elegance we ascribe to human behaviour.

Furthermore, such interactions may fail to present the **information content** of the data. The data produced is the superficial representation of the "actual contents" or the information that underlies it. In general, most commercial DBMSs make little attempt to extract this deep-seated abstract information. Advances in data modelling have helped to bridge this gap (see, for example, Chen 1976, Smith and Smith 1977, Roussopoulos 1977, Mylopoulos et al. 1980). However, the data models are tools meant principally for the database administrator. They provide little guidance to the user in interpreting the data. The task of interpretation and obtaining a "feeling" for the information content of the data still rests mostly with the user. A system such as the one discussed here transfers some of the responsibility of data interpretation from the user to the computer system. It undertakes a guided search of the data that satisfy the user's query and attempts to extract a brief qualitative expression describing the information therein.

Currently, while producing summary responses, the system stops as soon as any heuristic is successful in obtaining a pattern. Such responses are composed of one or more predicate forms, as explained in section 4.3. However, the first such response may not be the "best" possible one. In order to obtain the best answer, it is advisable to continue the process of identifying responses using the remaining heuristics. If, ultimately, several answers are obtained, a decision regarding which one to present to the user must be made. For this purpose, each answer could be assigned a weight and those with weights below a particular threshold would not be presented. Although the problem of assigning weights is encountered in several other applications of artificial intelligence, the issues involved are complicated; we do not delve into this topic here.

There are a number of issues that arise concerning the interaction of the data base and the knowledge base. The current system depends on the discovery of relationships occurring in the data base and makes use of the knowledge base only to find distinguishing values, possible joins, and appropriate heuristics. Since the heuristics are universal in nature, this implies that the techniques employed here can be transported to another domain (or used by another set of users) without undue modification. The only changes that have to be incorporated are new relation and attribute frames for each new database domain (or each new type of user).

Unfortunately, the portability is achieved by going directly to the data base, and is bought at the expense of using reasonably inefficient sequential searches (see section 5). This raises the question of whether it might be possible to avoid database access altogether. The current knowledge base is too impoverished to be used directly, but we could consider various enhancements. One possibility might be to generalize the idea of distinguishing values to provide rules describing the criteria for membership in a given class. For example, one such criterion could be that a passing mark is a grade point of 1 so that any question such as *Who failed CMPT 110?* could be answered with *All students with a mark of less than 1* without needing to consult the data base at all (assuming the user didn't know this so that the maxim of quality isn't violated). This kind of criterion would be simple to represent, but is obviously not a complete representation of what it means to fail a course since it ignores students who have failed by withdrawing too late, by not writing the final exam, by murdering the professor, etc. In general, to represent all the various subtleties of such criteria is a substantial problem in knowledge representation (consider, for example, having to represent the qualifications needed for a scholarship or all the requirements to get an undergraduate degree). Although it would be nice to be able to represent such general rules, it should be pointed out that consulting the data base as in our current approach circumvents the need to consider these representation issues. The heuristics can pick out relevant commonalities among students who failed a course, or won a scholarship, or received an undergraduate degree, without the need for sophisticated knowledge representation techniques.

Even if the representation issues were to be solved, however, database access would still be necessary. A prime reason is that any general rules will sometimes have exceptions that can only be discovered by looking at the data. Our heuristics currently allow some such exceptions to be found, although they are by no means a complete solution to the problem of exceptions. The modified equality and inequality heuristics (section 4.2.3) explicitly allow for occasional deviations (e.g., see Q6-S6), and the conjunction and disjunction heuristics can find characteristics common to entire exception classes (e.g., if one entire section of a class were given exemption from the final examination, the disjunction heuristic could answer the question *Who completed CMPT 378?* with the response *All students who wrote the final examination or were in section 02.*). Nevertheless, there are still open questions involving exceptions (such as being less ad hoc in defining exactly what *a few* means in the modified equality and inequality heuristics), which could be worked on as a further direction of this research.

There is still a third reason (besides avoiding knowledge representation problems and recognizing exceptions) that the data base should be consulted: to discover patterns in the data that can't be explicitly predicted. Consider the following question answer pair:

Q12: Which athletes failed HIST 101?

S12: The football players.

Response S12 summarizes information that can't be represented in a rule in the knowledge base (i.e., it isn't necessary that only the football players failed this course) and can only be found by looking through the data. Again, our heuristics would be able to find this pattern in the data, assuming that *football player* is a distinguishing value (which it might be to the athletic director, for example). In general there will be many such situations where the system knows that something interesting (i.e., something for which there is a distinguishing value) could occur in the data, but the exact context in which it actually occurs can't be foreseen.

Thus, it will normally be necessary to consult the data base. Nevertheless, an interesting research direction will certainly be to enhance the knowledge base as much as possible to provide rules that can at least direct the search through the data with more subtlety than distinguishing values are able to.

Another possible knowledge-base extension, which would avoid the problem of having rules disembodied from the data they reflect and which might be an answer to some of the efficiency problems of database consultation mentioned above, is to create a "memory" that would store patterns found in previous database searches. In other words, the memory would store the scalar implicatures that the system finds to be valid in the data base. This is similar in intent to Lebowitz's (1983) RESEARCHER system, which attempts to generalize concepts read from patent abstracts into a **generalization-based memory.** We would not be as concerned with describing how given instances differ from their generalizations, but *would* have to be concerned with how the generalizations change as the database contents are modified.

The memory would obviate the need to search the data base for repeated queries. For example, let a question Q that has been answered by the system have an answer A stored in an internal form. If the question Q is posed by the user again, the answer A can be returned. Similarly, if the question posed matches A, Q can be produced as the answer. For example, let us consider the example Q7-S7 from section 4.2.4. If the question posed is Q7-1, which is the interrogative form of S7, the answer provided may be S7-1, the assertive form of Q7.

Q7-1: Who are the students with cumulative GPA of 2.0 or less?

S7-1: All students who have been advised to discontinue studies at the University.

In some cases, it may not be possible to phrase a meaningful English question corresponding to the interrogative form of the response to a query. This is especially true in situations where complex responses are produced (e.g., in questions Q6, Q10, and Q11 in section 4). However, it may be possible to break up a complex query into two or more sub-queries. If answers to these sub-queries are already resident in the memory, the system may be able to compose the final response from the existing answers to these sub-queries. Clearly, the amount of search required to answer the query may be considerably reduced if parts of the answer can be retrieved from the memory, assuming the memory itself is organized for efficient retrieval.

The memory would usually be empty at system initialization; it would grow in size as the system interacted with the user and learned new facts about the data. It would have to be modified as the data in the data base changed. This would mean that the memory would somehow have to keep track of how the stored queries related to the data that produced them so as to be able to determine which queries would be affected by a given change in the data. It would also require some means of determining how new data affected queries summarizing existing data. This is the reverse process to that suggested by the Mays (1982) monitoring scheme, where monitors are posted to look for future changes in the data base. The memory part of our system would have to reason backwards from the current situation to infer how changes affect previously abstracted summaries. Whether Mays's temporal logic can be adapted to be useful in backwards reasoning is an interesting question. In any event, the amount of processing time required to keep the memory up to date is unclear. However, it would seem to be a computationally intense activity, which suggests there would be a trade-off between the time spent maintaining the memory, on the one hand, and the time saved in database access by having the memory, on the other hand.

As mentioned earlier, the database manager can implement different user models by creating different sets of attribute and relation frames for each type of user. This capability is similar to the idea of database *views*. For each type of user, the system would contain information about stereotypical knowledge possessed by that class of user. Different classes of users have different ideas about what values are "distinguishing" (e.g., an average of 4 might be fairly insignificant to an undergraduate accessing a student record data base, but to a graduate student it represents the dividing line between being allowed to graduate or not). The CUMULATIVE-GPA attribute frame in the graduate student user model would therefore be different from the CUMULATIVE-GPA frame in the undergraduate student model. For security purposes, it might be useful to prevent certain joins from taking place (e.g., it wouldn't be appropriate for students to access their professors' marks lists by joining the relation COURSE-REGISTRATIONS to the relation MARKS-LISTS (say) on the attribute COURSE-NO). The student user models could reflect this by appropriate restrictions on the relation frames. It is even possible to prevent summary responses altogether for certain attributes by having no distinguishing values in the corresponding attribute frames, or by providing them with a "nil" preference category. Such security and privacy considerations can be important for certain classes of users. All of this is currently possible, although not something we have actively experimented with. An extension to this capability might make it possible for the user to customize the kinds of summary responses he/she receives, rather than relying on the database manager to provide him/her with the appropriate user model. Whether to have the user fill in a template corresponding to each attribute frame, or whether to use natural language to specify the information in the various attribute frames is an open research question.

In the present system we have assumed that the system is provided with (and produces) a formal representation of the user's query. Ideally, the system's interface should include a natural language parser and generator, but as discussed earlier (section 4.3) this issue was not tackled here. There are still many open questions having to do with surface language, apart from issues of interpretation and generation per se. One such question of particular interest to this research is categorizing types of surface language that demand a summary response, as opposed to types that demand an extensional response or types where either an extensional or a summary response are appropriate. For example, *What are the characteristics of the students who failed CMPT 110?* requires a summary response; *Give me the names of the students who failed CMPT 110* demands an extensional response; and *Who failed CMPT 110?* allows for either kind of response. However, the problem is subtle. For example, the request *Give me the names of the students who registered on Wednesday* could be answered with an extensional response (which would normally be what is expected) or conceivably the summary response *Those with surnames beginning with the letters N through R.*

The key to recognizing what kind of response is needed is to recognize the user's intent (or at least his/her knowledge) in asking the question; that is, to consult a user model to see which kind of answer is appropriate. Thus, if it is known that the user is an administrator in charge of registration and that he/she is formulating registration policies, the second answer above might be reasonable. If the user is a clerk in charge of sending out registration forms, the first might be correct. Finally, if the user already knows all the names, then perhaps the summary response is desired (assuming the user has been unable to discern the pattern on his/her own).

The kind of user model needed to handle this is more sophisticated than the simple user model currently used. To see this, let's look at the ambiguous query *Who failed CMPT 110?* once again. This question can admit either a summary response or an extensional response. If the system knows the user knows all the students who failed CMPT 110, then some description of their characteristics (e.g., *students who were absent from the final examination*) is probably more appropriate. On the other hand, if the system knows the user knows that students who miss the final examination fail the course, then a summary response describing this fact would be inappropriate, and a list of the students' names is likely what is desired. This won't be foolproof, of course. The user could be asking for a re-iteration of something he/she already knows (for confirmation purposes, perhaps) or could be asking for another summary pattern besides the one the user already knows. Another subtlety that arises is the distinction between implicit and explicit knowledge – the user may know something but not realize it, or may not be able to make the inferences needed to deduce something that he/she has the knowledge to deduce. For example, the user may know the names of all the students who failed CMPT 110 but not realize these are the only students; or he/she may know everybody who didn't write the final examination and also the rule that if the final examination is missed a student fails the course, but the user may not have applied the rule in this case. Finally, for some extensional responses, it still might be appropriate to repeat a general rule that the user knows, just to re-inforce in his/her mind the applicability of the rule in this situation. Thus, if the user has asked which managers earn more than $40K (see Q2), then even if the user knows that in general all managers earn over $40k, it might be useful to re-iterate this fact after producing the list of managers' names since it would be difficult for the user to check that all the names had appeared without exceptions (especially if the list were long).

These kinds of complications make the task of devising the user model quite tricky. It must keep track of subtle degrees of knowledge, incomplete knowledge, changing knowledge, laziness in applying knowledge, etc., and it must be possible to recognize user's intentions in the use of this knowledge. There is a growing body of research involved in representing the kinds of knowledge needed here, and in dealing with language as intentional behaviour. The work by the University of Toronto group, in particular, pioneered this approach (see Cohen 1978, Allen 1979, and Allen and Perrault 1980, for example) and could form a starting point for research into user model extensions. The first problem would be to represent what the user knows and doesn't know (since many of the decisions about what to present to him/her depend on this). Subsequent steps could get into recognizing intentions and other sophisticated discourse phenomena.

There are other, more subtle, problems that arise with this approach to summary response generation. One such problem involves avoiding the production of responses that "overlap" (i.e., are implicit in) the question. Such overlapping definitions themselves violate Grice's maxims of relation and quantity. For example,

Q14: Which students had a GPA of greater than 5?
S14: All students with a GPA of greater than 5.

or

Q15: Which graduate students are both teaching and research assistants?
S15: All graduate students receiving money for teaching and being paid by a professor to do research.

Simple cases like Q14-S14 can be prevented by explicitly prohibiting responses that have the same predicates as the question. This would apply even if only one conjunct or disjunct is the same in both question and response. Q15-S15 presents a more complex problem since the answer, although directly implied in the user's mind, may in fact involve attributes different from the question (e.g., the data base may have attributes such as TEACHING-ASSISTANT?, RESEARCH-ASSISTANT?, MONEY-RECEIVED-FROM-TEACHING, MONEY-RECEIVED-FOR-RESEARCH). In cases such as this, where the data base is in some sense redundant, extra information would have to be added to the attribute frames to indicate overlapping attributes. This information could then be used to avoid producing responses that overlap the query. Even this extension would not provide a total solution to the problem, since the user may be able to make many subtle connections among the data in the data base that will lead to an overlapping response from his/her point of view. Additional user modelling techniques to those discussed above will have to be developed to predict these connections and thus prevent the production of a response implicit in the query.

Another subtle problem that arises is the problem of "accidental summaries", i.e., summaries that are true of the current data base but not in general. Our use of distinguishing values is an attempt to reduce the chances of this occurring, but it can still happen. For example, it may be true in the simple student data base that, currently, all people who are from Canada also have NSERC grants. "NSERC" may also be a distinguishing value for the NATURE-OF-FINANCIAL-AID attribute (e.g., to answer question Q3). However, to respond to the question *Who are the students from Canada?* with the answer *All students with NSERC grants* might mislead the user into thinking that there was some necessary connection between being from Canada and having an NSERC grant, rather than an accidental one. Accidental summaries violate Grice's maxim of quality in that they imply something is true that is not. Just avoiding the production of summary responses in such cases will not solve the problem, since it still may be very useful to produce a summary response. Thus, it may be accidental that all managers earn over $40k, but answer S2-1 (*Abel, Baker, Charles, Doug.*) to question Q2 (*Which department managers earn over $40k per year?*) still (normally) violates Grice's maxims, and the summary response S2-2 (*All of them.*) is still (normally) more appropriate. The only long term solution to this problem is to expand the knowledge base with further information about necessary relationships in the world being modelled (e.g., for the student data base, the knowledge base could be stocked with rules and regulations about academic programmes, student eligibility for various prizes, etc.) These necessary relationships could then be used to clarify the summaries provided to the user as to whether accidental or necessary relationships were being reported.

In conclusion, we would like to say that, despite its use over the last twenty years, the database environment still forms a nice microworld to study a variety of natural language issues. Hopefully, some of these have been illuminated by this research.

7 Acknowledgements

This paper is based on graduate research done by Jugal Kalita at the University of Saskatchewan. We would like to thank the other members of his M.Sc. thesis committee, Robin Cohen and Paul Sorenson, for their constructive comments. Thanks also to the referees for their useful and constructive comments – that the final version of this paper is considerably better than the original is, in no small measure, due to them. We would also like to acknowledge the financial support of the University of Saskatchewan and the Natural Sciences and Engineering Research Council of Canada.

References

Allen, J.F. 1979 A Plan-Based Approach to Speech Act Recognition. TR-131, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Allen, J.F. and Perrault, C.R. 1980 Analyzing Intention in Dialogues. *Artificial Intelligence* 15(3): 143-178.

Codd, E.F.; Arnold, R.S.; Cadious, J.M.; Chang, C.L.; and Roussopoulos, N. 1978 RENDEZVOUS Version 1: An Experimental English Language Query Formulation System for Casual Users of Relational Databases. Research Report No. RJ2144 (29407), IBM Research Laboratory, San Jose, California.

Chen, P.P.S. 1976 The Entity-Relationship Model – Towards a Unified View of Data. *ACM Transactions on Database Systems* 1(1): 9-36.

Cohen, P.R. 1978 Planning Speech Acts. TR-118, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Davidson, J. 1982 Natural Language Access to Database: User Modelling and Focus. *Proceedings of Fourth National Conference of the Canadian Society for Computational Studies of Intelligence,* Saskatoon, Saskatchewan, Canada: 204-211.

Grice, H.P. 1975 Logic and Conversation. In Cole, P. and Morgan, J.L., Eds., *Syntax and Semantics: Speech Acts,* Vol. 3. Academic Press, New York: 41-58.

Grosz, B.J. 1977 The Representation and the Use of Focus in a System for Understanding Dialogues. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.* Cambridge, Massachusetts: 67-76.

Harris, L. 1977 User Oriented Database Query with the ROBOT Natural Language Query System. *International Journal of Man-Machine Studies* 9: 697-713.

Janas, J.M. 1979 How to Not Say "NIL" – Improving Answers to Failing Queries in Data Base Systems. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence.* Tokyo, Japan: 429-434.

Joshi, A.K.; Kaplan, S.J.; and Lee, R.M. 1977 Approximate Responses from a Data Base Query System: An Application of Inferencing in Natural Language. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.* Cambridge, Massachusetts: 211-212.

Kalita, J.K. 1984 Generating Summary Responses to Natural Language Database Queries. Technical Report 84-9, Department of Computational Science, University of Saskatchewan, Saskatoon, Canada.

Kalita, J.K.; Colbourn (Jones), M.J.; and McCalla, G.I. 1984 A Response to the Need for Summary Responses. *Proceedings of COLING-84: 10th International Conference on Computational Linguistics.* Stanford, California: 432-436.

Lebowitz, M. 1983 RESEARCHER: An Overview. *Proceedings of the American Association for Artificial Intelligence Conference (AAAI-83).* Washington, D.C.: 232-235.

Kaplan, S.J. 1982 Cooperative Responses from a Portable Natural Language Query System. *Artificial Intelligence* 19(2): 165-187.

Lee, R.M. and Gerritsen, R. 1978 Extended Semantics for Generalization Hierarchies. *Proceedings of the 1978 ACM SIGMOD International Conference on Management of Data.* Austin, Texas.

Mays, E. 1982 Monitors as Responses to Questions: Determining Competence. *Proceedings of the American Association for Artificial Intelligence Conference (AAAI-82).* Pittsburgh, Pennsylvania: 421-423.

McCoy, K.F. 1982 Augmenting a Database Knowledge Representation for Natural Language Generation. *Proceedings of the Twentieth Annual Conference of the Association for Computational Linguistics,* Toronto, Ontario, Canada: 121-128.

McDonald, D. 1983 Natural Language Generation as a Computational Problem: An Introduction. In Brady, M. and Berwick. R., Eds., *Computational Models of Discourse,* MIT Press, Cambridge, Massachusetts: 209-265.

McKeown, K.R. 1982 The TEXT System for Natural Language Generation: An Overview. *Proceedings of the Twentieth Annual Conference of the Association for Computational Linguistics,* Toronto, Ontario, Canada: 113-120.

Mylopoulos, J.; Bernstein, P.A.; and Wong, H.K.T. 1980 A Language Facility for Designing Database-Intensive Applications. *ACM Transactions on Database Systems* 5(2): 185-207.

Mylopoulos, J.; Borgida, A.; Cohen, P.; Roussopoulos, N.; Tsotsos, J.; and Wong, H.K.T. 1976 TORUS: A Step Towards Bridging the Gap between Databases and the Casual User. *Information Systems* 2(1): 49-64.

Plath, W.J. 1976 REQUEST: A Natural Language Question Answering System. *IBM Journal of Research and Development* 20(4): 326-335.

Reiter, R.; Gallaire, H.; King, J.J.; Mylopoulos, J.; and Webber, B.L. 1983 A Panel on AI and Databases. *Proceedings of the 8th International Joint Conference on Artificial Intelligence.* Karlsruhe, West Germany: 1199-1206.

Roussopoulos, N. 1977 A Semantic Network Model of Databases. TR104, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Sidner, C.L. and Israel, D.J. 1981 Recognizing Intended Meaning and Speakers' Plans. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence.* Vancouver, British Columbia, Canada: 203-208.

Small, S. 1980 Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding. TR-954, University of Maryland, College Park, Maryland.

Smith, J.M. and Smith, D.C.P. 1977 Database Abstraction: Aggregation and Generalization. *ACM Transactions on Database Systems* 2(2): 105-133.

Thompson, B.H. and Thompson, F.B. 1975 Practical Natural Language Processing: The REL System as Prototype. In: Rubinoff, M. and Yovits, M., Eds., *Advances in Computers* 13. Academic Press, New York: 109-168.

Waltz, D.L. 1978 An English Language Question Answering System for a Large Relational Database. *Communications of the ACM* 21(7): 526-539.

Webber, B.L. 1978 Description Formation and Discourse Model Synthesis. *Proceedings of the Theoretical Issues in Natural Language Processing Workshop (TINLAP-2),* University of Illinois at Urbana-Champaign, Illinois: 42-50.

Woods, W.; Kaplan, R.; and Nash-Webber, B. 1972 The Lunar Sciences Natural Language Information System: Final Report. BBN Report 2378, Bolt, Beranek and Newman Inc., Cambridge, Massachusetts

Note

1. UNIX is a trademark of AT&T Bell Laboratories.