# Towards More Graceful Interaction: a Survey of Question-Answering Programs.

CUCS-209-85

Cecile L. Paris

Department of Computer Science
Columbia University
New York City, New York, 10027 USA

## Abstract

One of the aims of Natural Language Processing (NLP) is to facilitate the use of computers by allowing users to interact with systems in natural language. Since such interactions often take the form of question-answering sessions, the process of question-answering is an important part of NLP. In this paper, we are concerned with the progress made towards building question-answering systems which are natural and satisfying to users, allowing them to express themselves freely and answering questions appropriately. Such systems are said to provide graceful interaction. We survey the evolution of question-answering programs, presenting steps and approaches that have been taken towards providing graceful man-machine interaction, and pointing out eficiencies of existing programs and theories. Our emphasis is on the various issues and difficulties encountered in building a user-friendly question-answering program.

## Table of Contents

## List of Figures

## 1. Introduction

One of the aims of Natural Language Processing (NLP) is to facilitate the use of computers by allowing users to interact with systems in natural language. Since such interactions often take the form of question-answering sessions, the process of question-answering is an important part of NLP. In this paper, we are concerned with the progress made towards building question-answering systems which are natural and satisfying to users, allowing them to express themselves freely and answering questions appropriately [20]. Such systems are said to provide *graceful interaction.*

### 1.1 Components of graceful interaction

The following criteria have been identified in the literature as important facets in the overall process of graceful interaction for question-answering systems [20, 69].

**Flexible input:** the system should deal meaningfully with anything the user may reasonably ask. To do so, a wide syntactic and semantic coverage as well as the handling of ellipsis and the resolution of anaphora is necessary. Features such as spelling correction and synonym capabilities can further enhance the interaction.

**Data-independence:** the system should free the user from studying the contents and structure of the knowledge base before asking questions.

**Cooperation:** the system should provide cooperative answers. In asking a question, the user wants to obtain some information. A complete answer is one that will provide such information in a coherent manner, and not necessarily one that literally answers the question.

**Avoidance of confusion:** the system should avoid misleading the user. Users have beliefs about the knowledge base when asking questions. When their beliefs contradict facts in the knowledge base, a system should make sure that answers given do not reinforce these beliefs.

The user expects question-answering programs to exhibit answering capabilities commensurate with their apparent understanding abilities; i.e. the user expects a system to provide meaningful and cooperative answers, as a human would. So, while it is possible to provide more graceful interaction with a program by adding human engineering features (as in LIFER [21] and PLANES [73]), implementing a truly cooperative system requires studying how people communicate with each other; this leads to the development of theories of human question-answering behavior [36], [44], [30, 32], [40], [1].

The emphasis of this paper is on the various issues and difficulties encountered in building a user-friendly question-answering program. We survey the evolution of question-answering programs, presenting steps and approaches that have been taken towards providing graceful man-machine interaction, and pointing out deficiencies of existing programs and theories.

### 1.2 Overview of the paper

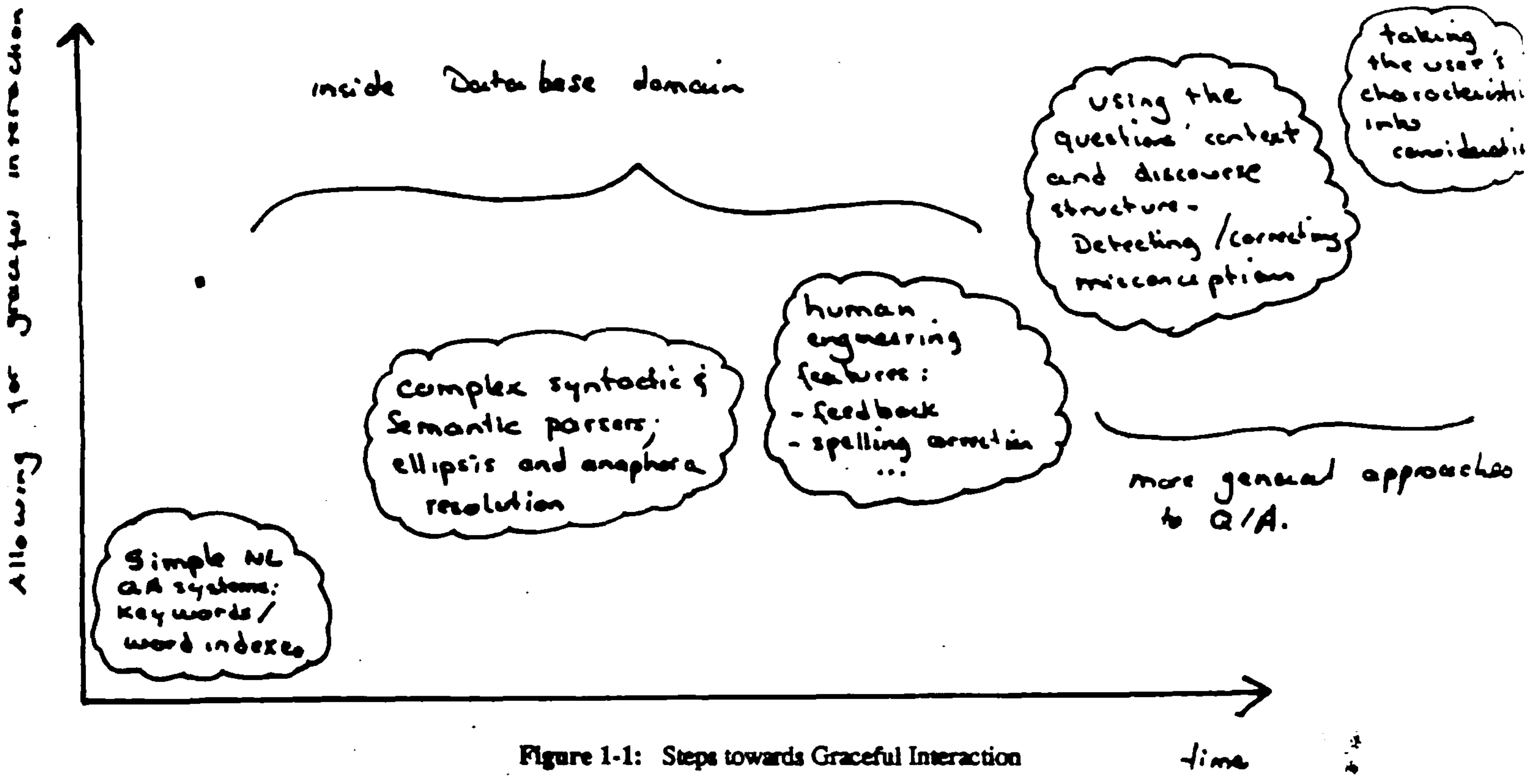
The desire to have easily accessible database systems prompted the initial research into natural language interfaces. This work gave rise to the first question-answering programs: BASEBALL [14], SYNTHEX [66], LUNAR [77], LIFER [21] and PLANES [73]. These first natural language interfaces were aimed at allowing users to pose questions to a database in English in as natural a way as possible. Complex syntactic and semantic parsers were developed to permit users to express themselves using a variety of English constructions (LUNAR). Anaphora and ellipsis resolution were added to allow users to state their questions more concisely (LUNAR, LIFER and PLANES). Efforts were made to render man-machine communication as comfortable and natural to the user as possible by augmenting systems with features like spelling correction and paraphrasing the question (first in LIFER, PLANES, and, later in CO-OP [31, 43]). Systems with such complex parsers and additional features are presented in Section 2.3.

Following these initial developments, the process of question-answering was studied in a more general framework [36, 24, 1]. Researchers analyzed the factors involved in understanding a question and answering it appropriately. By studying how humans use natural language, researchers were able to develop systems exhibiting cooperative behavior, such as:

- understanding a question within its context to answer it more appropriately [36, 62]
- providing more information than actually asked for [36, 1, 29]
- detecting and correcting misconceptions on the part of the user [32, 40, 41]
- allowing the user to ask questions about the database structure [45]

These systems and the theories they embody are presented in Section 3.

Finally, knowledge of the user was also found to be an important factor in question-answering [1, 53, 11]. Methods for exploiting such knowledge to further enhance the capabilities of question answering systems are presented in Section 4. The chart in Figure 1-1 plots the steps that have been made towards graceful interaction in man-machine interaction.



Allowing for graceful interaction
inside Database domain
Simple NL Q/A systems: Keywords / word indexes
complex syntactic & Semantic parsers; ellipsis and anaphora resolution
human engineering features: - feedback - spelling correction ...
using the questions' context and discourse structure - Detecting / correcting misconceptions
taking the user's characteristi into considerati
more general approaches to Q/A.
time

Figure 1-1: Steps towards Graceful Interaction

## 2. Natural language interfaces to databases

Many of the early investigators of natural language question-answering systems implemented front-ends for database systems. This type of program was widely studied for two primary reasons. First, naive and casual users frequently found the task of learning a database query language both painful and tedious. (Naive users are those who do not know much about a computer system and casual users those who do not use one often). As these users were numerous, there was a clear need for simplified access to databases. By allowing users to pose queries in English (even if restricted), a question-answering program provides easier access to databases than database query languages do. Second, the widespread use of databases and their availability offered a convenient testbed for studying and evaluating natural language interfaces.

Database query languages provide a concise means for specifying retrieval requests. While they are unambiguous, they are not always easy to use: just as one has to learn a particular programming language syntax in order to be able to write a program in that language, a database user has to learn a specific query language. As an example, a user unfamiliar with ISBL (Information System Base Language) [72] will not be able to issue the query shown in Figure 2-1. Moreover, the user needs to know what kinds of knowledge are available and how it is organized.

```
In ISBL (Information System Base Language),
a query which prints the names of members with
negative balances is expressed as follows:

        LIST MEMBERS: BALANCE < 0 % NAME
```

**Figure 2-1:** Example of a query in a database query language

Seeking easier communication and more freedom of expression for the user, researchers started to study the problems of natural language interfaces to database systems.

### 2.1 Early question-answering programs

BASEBALL and SYNTHEX were among the first natural language question answering programs. These two programs used very different approaches to parse questions and find answers: BASEBALL searched for keywords while SYNTHEX used inverted indices. BASEBALL answered questions about American League baseball games in a given year and SYNTHEX had a database of English text (the *Golden Book Encyclopedia*). The domains of these programs were quite restricted, however, so their designers were able to ignore many of the issues involved in understanding a question and answering it adequately. The use of toy databases built solely for testing purposes further simplified the development of these programs.

#### 2.1.1 The keyword approach: BASEBALL

BASEBALL [14] grouped the elements of a question into functional phrases and transformed them into canonical expressions called *specification lists* (Figure 2-2). The database was then searched for items matching that list exactly. Because there was no inferencing mechanism, no answer to the question was found if no item in the database matched the specification list exactly, even if the answer could theoretically be derived.

In BASEBALL, the question:

*On how many days in July did eight teams play?*

would be grouped in the following way:

*(on how many days) (in july) did (eight (teams)) play?*

and give rise to the specification list:

*day (number of) = ?*
*month = july*
*team (number of) = 8*

**Figure 2-2:** Example of a query in BASEBALL

## 2.1.2 The inverted index approach: SYNTHEX

SYNTHEX [66] used a textual database and some clever indexing techniques to retrieve facts from the database. Essentially, an inverted index reference was made for every word in the text, and every word in the question was mapped back into the text. A scoring mechanism was then used to choose the facts that constituted the answer, based on the completeness of the syntactic agreement of the sentence found in the text and the question. Figure 2-3 shows a question, sentences found in the database using the inverted index mechanism, and the syntactic agreement of these sentences with the question.

For the question: *What do worms eat?*

The following facts might be found:

1- Facts in complete syntactic agreement: *Worms eat grass*
*Grass is eaten by worms*

2- Facts in partial agreement: *Worms eat their way through the ground*
*Horses with worms eat grain*

3- Fact with no agreement: *Birds eat worms*

**Figure 2-3:** Use of inverted indices in SYNTHEX

This approach had the advantage of being easily adaptable to other domains, since it required only the creation of an inverted file. Unfortunately, it also had the major problem of being based solely on words, and not on their meanings (or the concepts they represent). As a result, it was unreliable, incomplete, and could not be considered to really "understand" a question.

### 2.1.3 Summary

In these two early approaches to question-answering, very little syntactic analysis was done, and semantics were taken into consideration only insofar as they were embedded in the patterns and the heuristics used in interpreting questions. Searching for keywords and patterns does not, in most cases, adequately capture the intended meaning of a question. Moreover, this method of parsing imposes a burden on the user: a question can be expressed in English, but, in order to use the program efficiently, the user must know what types of patterns (or keywords) are recognized and what is contained in the database. Adding patterns (keywords) to the parser will increase the probability that a query entered by the user will be meaningful to the system. Still, most inputs would not be recognized; building a system having reasonable generality would require an exorbitant number of patterns. Moreover, these systems lacked the deductive power necessary to provide answers which could be inferred (deduced) from facts in the database but were not explicitly stated therein.

## 2.2 More complete natural language interfaces

The drawbacks of the earliest question-answering programs made it clear that a more careful analysis of the input was necessary in order to understand natural language (whether texts or questions). Subsequent research efforts were directed towards studying syntax and semantics. Good syntactic and semantic parsers together with large dictionaries achieve a large linguistic coverage, allowing users to express themselves more freely. Furthermore, users can communicate with the machine in a more concise and natural manner if they are able to use anaphora (such as pronouns) and ellipsis[1].

A major step in natural language interfaces was the development of such good syntactic and semantic parsers and the treatment of anaphora and ellipsis. LUNAR was the first program that dealt in a comprehensive way with both syntax and semantics. Later, programs were augmented with human engineering features (such as spelling correction and clarification dialogs) aimed at easing the man-machine interaction. We present LIFER and PLANES as examples of such programs.

### 2.2.1 Broad linguistic coverage and resolution of anaphora and ellipsis: LUNAR

Designed as an aid to lunar geologists, LUNAR [77] was the first natural language interface to an existing database. The database described moon rock samples. The geologists could ask for samples with given attributes to be retrieved (see Figure 2-4). Users could express themselves in a fairly unrestricted manner, use embedded clauses, refer to items by descriptions rather than by tags ("sample containing phosphorus" instead of "S10024"), and make use of anaphoric references and pronouns. The questions they were allowed to ask, however, were restricted to questions that could be translated into database queries.

LUNAR had a large dictionary and a powerful augmented transition network (ATN) syntactic parser [76] which could handle many of the subtleties of Englisn grammar in the limited context of a database. Input sentences were parsed to produce a parse tree. This parse tree was then examined by a semantic analyzer, which related the words and the syntactic structures in which they occurred to concepts in the database and the relationships between these concepts. In LUNAR (and its predecessor, Airline Guide [75], whose semantic parser served as a basis for that of LUNAR), the semantics were represented as procedures. With this representation, LUNAR was able to deduce facts

[1]An ellipsis is the omission of some words in a sentence.

1) Some questions handled by LUNAR:

- Give me all the lunar samples with magnetite.
- Which samples are breccias?
- Of the type A rocks which is the oldest?

2) Attachment of prepositional phrase:

Give me all analyses of sample 10046 for hydrogen.

''For hydrogen'' is a prepositional phrase. Syntactically, it could be attached to either ''analyses'', ''sample 10046'' or ''give''. LUNAR is able to attach it to the appropriate noun head, ''analyses'', by backtracking between the syntactic and the semantic parsers.

3) Example of an ellipsis:

The following sequence

1) Give me all analyses of sample 10046 for hydrogen.
2) for oxygen.

is more natural than:

1) Give me all analyses of sample 10046 for hydrogen.
2) Give me all analyses of sample 10046 for oxygen.

**Figure 2-4:** Example of questions handled by LUNAR

not explicitly mentioned in the database. Even though the syntactic and semantic parsers were not fully integrated, backtracking between the two allowed LUNAR to attach prepositional phrases to the appropriate head noun (see example 2 in Figure 2-4). The output of the semantic parser was a query expressed in the language of the underlying database. Finally, the database system received control and processed the query in the conventional manner.

LUNAR's ATN parser could handle tense, modality, adjective modifiers, embedded constructions and relative clauses. Furthermore, to allow more freedom of expression, LUNAR handled anaphoric references and ellipsis (see Figure 2-4). To solve anaphoric references, LUNAR kept a reference list of all noun phrases. That list was searched for a noun phrase whose syntactic and semantic structures matched those of the anaphoric reference. That noun phrase was considered to be the antecedent. To handle ellipsis, the current query was compared with the previous one. If the syntax and semantics matched, the missing part would be taken from the previous query. Because anaphoric references and ellipsis allow the user to express himself in a more concise and natural fashion (Figure 2-4), it is important for natural language systems to be able to resolve them. Finally, the representation of semantics as procedures in LUNAR was a significant development in NLP, since these procedures were able to deduce facts not explicitly mentioned in the database. Consequently, LUNAR could answer a greater range of questions than would have been possible otherwise.

With its powerful ATN parser and large dictionary, LUNAR provided a large linguistic coverage and allowed for a greater range of questions. Such qualities made LUNAR a successful natural language interface[2]. Broad linguistic coverage as well as resolution of ellipsis and anaphoric reference became fairly standard features in subsequent natural language interfaces, and the focus of research shifted to additional human engineering issues.

### 2.2.2 Tolerance of spelling errors and synonym capabilities: LIFER

An interface can be rendered more user friendly by adding a few human engineering features. In LIFER, for example, a spelling correction mechanism attempted to understand input containing spelling errors, users were allowed to define their own terminology (in terms of the system's terminology), and, to minimize misunderstanding between the system and the user, the user was kept informed of the system's state of processing.

LIFER [21] was a tool for easing the construction of natural language interfaces to databases. Two applications were developed for LIFER: the LADDER system [55], which answered questions about a naval command and control database, and a database system with information that described a department in a university.

LIFER had a *semantic grammar* parser that tried to match an input sentence against templates based on *semantic categories* (see Figure 2-5). When a template was matched, the corresponding database query was issued.

The template: what is the *<attrib>* of the *<ship>*

(where *<attrib>* and *<ship>* are semantic categories) would match:

what is the *length* of the *Constellation* ?
what is the *displacement* of the *Nautilus*?

Both *Constellation* and *Nautilus* are marked in the database as being *ships*, and both *length* and *displacement* are marked as being *attributes*.

Parse tree produced:

```
                        TEMPLATE 1
      --------------------------------------------
      |      |     |         |         |      |      |
     WHAT   IS    THE   <ATTRIBUTE>    OF    THE   <SHIP>
      |      |     |         |         |      |      |
      |      |     |         |         |      |      |
     what   is    the      length      of    the   <SHIP-NAME>
                                                     |
                                                     |
                                                 Constellation
```

Figure 2-5: Example of a template in LIFER

[2]An evaluation [69] showed that LUNAR was able to answer about 90% of the questions geologists posed (after a few minor modifications). Note, however, that questions tended to be relevant to the database.

To handle anaphoric references and incomplete input, LIFER tried to match the current utterance with part of the previous question's parse tree if the initial matching process against templates failed. Upon success, the fragment matching from the previous query was inserted into the current tree, and the new query generated, as shown in Figure 2-6.

---

The template matched: *What is the <attrib> of the <ship>*

*What is the homeport of the Constellation?*

If the following question was:

*of the Nautilus?*

it would be correctly interpreted as

*What is the homeport of the Nautilus?*

**Figure 2-6:** Ellipsis resolution in LIFER

---

Note however, that the fragment "*for* the Nautilus" could not be understood, since the template previously matched contained the preposition "*of*". Such behavior is understandable when one knows how the ellipsis resolution is done, but inconsistent and incomprehensible for a naive user. This detail highlights a major difficulty in making systems that are natural to users: the system's expressive capabilities are limited in comparison with the richness of a natural language. There are always constructions and vocabulary that the system does not understand. Unfortunately, these limits are hidden to the user, until they are stumbled upon[3]. At this point, however, it is typically very difficult to understand what happens. The user will not necessarily think about rephrasing his query, but may attempt to explain to the system what was meant. This behavior comes about because the naive user, after observing the system's apparent comprehension of natural language, does not realize that the system, unlike a human being, only understands a limited subset of the language. Such an interaction will cause even more confusion as the system will probably not be able to understand such an explanation. Finally, because of the use of templates, there is no logical explanation for understanding one phrase and not another one, making it difficult for the user to discover what subset of English is understood by the LIFER system.

Spelling correction

LIFER attempted to understand input containing spelling errors. When a template matched except for one word, LIFER found possible candidates for that word, based on the word and the expected semantic category, as indicated by the template. The semantic categories were thus important to reduce the number of alternatives. The best match among those was then chosen (see interaction 1 in Figure 2-7). Spelling correction is not a major issue in question-answering. Yet, it is a nice feature in a question-answering program, for it can spare frustration to users by not making them retype questions because of small spelling errors. It is therefore a small step towards more pleasant interaction.

[3]Some programs use a menu to constrain the user's choice at any point and make the system's limits explicit [71]. This type of interface is, however, not very natural.

```
    1- What is the avarage salary and age for math department
       secretaries?
                        AVERAGE <== Spelling
    PARSED!
       ...

    2- for CS department secretaries?
    Trying Ellipsis: 1- What is the average salary and age for CS
                          department secretaries?
       ...

    3- What assistant professors in CS were hired in 77?
       ...
```

**Figure 2-7:** Spelling correction and ellipsis resolution in LIFER

Synonyms for individual words and whole sentences
To extend the capabilities of the system and adapt it to the user, LIFER allowed the user to define synonyms for both individual words and whole sentences. For example, in order to use *Compsci* instead of *CS*, the user could tell LIFER that they were synonyms (see interaction 25 in Figure 2-8). Similarly, the user could define his own phrasing for a sentence, using what Hendrix called the *paraphrase* feature, as in:

*let "describe John" be "print the height, weight, and age of John"*

Both these features are useful, since they allow users to tailor the system to their taste in certain ways. In order to do so, however, a user first has to know the exact syntax used for these features.

Feedback
Hendrix considered feedback an important humanizing factor. LIFER constantly informed the user of the state of processing (see interactions 1 and 2 in Figure 2-7): the user was informed when the sentence parsed successfully ("PARSED!"); when spelling correction or ellipsis substitution was done, an appropriate message was printed.

Another facet of the feedback feature was to allow users to see what went wrong and give them the opportunity to correct problems when inputs were not parsed correctly. When both the spelling error correction and the ellipsis substitution failed, LIFER told the user where the problem occurred, i.e., what word was not understood. It also specified what was expected, i.e. the semantic category indicated by the template (see interactions 20 and 24 in Figure 2-8). From there, the user could correct the question (interaction 23 in Figure 2-8). Finally, LIFER allowed a casual user to ask about the definition of some symbols (or semantic category), as in interaction 21 of Figure 2-8. Note that this type of interaction is also useful to train the user who can readily see what questions are understood by the system.

Note, however, that LIFER only told the user whether the question was parsed or not and did not show him what it understood the question to be. It can be argued that presenting a paraphrase of the question to the user to permit verification and correction of the query issued is a useful tool. In fact, paraphrasing can ensure the user that a system understood a query as it was intended before searching the database for the answer. (This approach has been implemented in TQA [50, 51, 10], PLANES [73], RENDEZ-VOUS [5, 6], CO-OP [31, 43], and other programs.)

```
20- What assistant Professors in compsci were hired after 1975
Trying ellipsis: Ellipsis has failed
THE PARSER DOES NOT EXPECT THE WORD "COMPSCI" TO FOLLOW
"WHAT ASSISTANT PROFESSORS IN".
Options for the next word or meta-symbol are:
<DEPARTMENT-NAME> OR <DEPARTMENT>

21- What is <DEPARTMENT-NAME>
PARSED!
<DEPARTMENT-NAME> may be any member of the set:
      [ ANTHRO ANTHROPOLOGY ART BS BUSINESS ... CS ...
        ZOO ZOOLOGY ]

<DEPARTMENT-NAME> may be any sequence of words
      following one of the patterns:
      <DEPARTMENT-NAME> => BUSINESS ADMINISTRATION
                           COMPUTER SCIENCE
                           ...

23- Use CS for Compsci in 20
PARSED!
(ID 263-42-6062 POSITION Assist-Prof DATE-HIRED 8/1/76)
(ID 501-13-1171 POSITION Assist-Prof DATE-HIRED 6/15/76)

24- How many associate professors are there in the
    compsci  department?
Trying ellipsis: Ellipsis has failed
THE PARSER DOES NOT EXPECT THE WORD "COMPSCI" TO FOLLOW
"WHAT ASSISTANT PROFESSORS IN".
Options for the next word or meta-symbol are:
<DEPARTMENT-NAME> OR <DEPARTMENT>

25- Define compsci like CS
PARSED!
COMPSCI
```

**Figure 2-8:** Synonym capability in LIFER

Limitations

LIFER allowed a good deal of flexibility but had consistency problems. Consider interactions 20-25 in Figure 2-8. In those interactions, LIFER was given a word it did not recognize. LIFER indicated this fact and told the user what was expected (inputs 20 and 21). The user was then able to correct the faulty word and reparse the question (input 23). At this point, the user asked the system to use "CS" instead of "COMPSCI". Essentially, he asked the system to consider "CS" and "COMPSCI" as synonyms. It would also be reasonable for the user to believe that, now, he could keep on using "COMPSCI" instead of "CS". This would certainly be the case in a human interaction, unless this substitution was a one time substitution (as in the case of a spelling error correction). However, this was not the case in LIFER: as long as the user did not define "CS" and "COMPSCI" as synonyms using the *define* option, he was unable to use these two words interchangeably. This is unintuitive and inconsistent.

Finally, because the template system and the juxtaposition method (Figure 2-9) used in LIFER did not provide as general a parsing mechanism as that employed by LUNAR, the system builder had to be very careful in constructing the templates and juxtaposition rules to make sure most of the possible questions and juxtapositions would be

Juxtaposition in LIFER was done by adding expansion rules for each juxtaposition allowed. For example, the rule:

*attribute -> <attribute> <attribute>*

would allow the following query:

*What is the length, displacement, and home port of the Constellation?*

Note that this rule is specific to *<attribute>* and cannot be used to join other semantic categories.

**Figure 2-9:** Expansion rules in LIFER

covered. Inconsistency problems and confusion for the user would arise if some possible constructs had been overlooked. Thus, LIFER worked best when the system builder was well informed about linguistics and the database.

## 2.2.3 Tolerance of ungrammatical inputs; clarification dialog and paraphrase: PLANES

While LIFER tried to be tolerant of spelling errors, one of the primary goals of PLANES was to be tolerant of grammatical mistakes. PLANES [73] was a natural language interface to a large database containing maintenance and flight records of naval aircrafts. PLANES was based on the assumption that the sentence-level structure of the input would not always be syntactically well formed, although the semantic structure would. As a result, PLANES made little use of syntactic information. Like LIFER, PLANES used a semantic analyzer. This analyzer was an ATN with one top level node that repeatedly called subnets, each of which identified a particular semantic constituent.

This method proved successful at parsing ungrammatical inputs (since grammar was ignored), thus easing interaction by allowing users to pay less attention to the syntactic structure of their questions and not worry about what syntactic constituents to use to be understood by the system. It had its own problems, however. Because all the constituents were parsed independently, there was no mechanism to choose a word sense depending on what had already been parsed. PLANES thus relied heavily on its domain to eliminate problems of word sense selection. This would sometimes lead to confusion. Second, this parsing technique caused modifiers to be parsed independently of the head noun they modified, thus defeating the purpose of a modifier (or prepositional phrase). These problems imposed constraints on the type of syntax the user could employ (which is contrary to the design goal of PLANES), and consequently on the user's freedom of expression, for he had to know what term to use when. Like LUNAR and LIFER, PLANES was able to handle anaphoric references and pronouns, using the previous query to complete a sentence. If the sentence could not be parsed, even after trying pronoun substitution, a *clarification dialog* would be initiated. Like LIFER, PLANES tried to explain what part of the input question failed. PLANES gave more feedback to the user than LIFER did by paraphrasing the question to make sure the generated database query was the one the user intended. After seeing the paraphrase, the user had the choice of accepting it or changing it. Although paraphrasing may provide useful feedback to the user, it may also get tedious, as illustrated in the hypothetical dialog presented in Figure 2-10.

```
U: Give me a list of all parts supplied by a company in Tennessee
S: I UNDERSTAND THE QUERY AS:
              PRINT ALL PARTS WHOSE SUPPLIER IS IN TENNESSEE
   IS THIS RIGHT?
U: Yes
S: <answer>

U: by a company in Texas
S: I UNDERSTAND THE QUERY AS:
              PRINT ALL PARTS WHOSE SUPPLIER IS IN TEXAS
   IS THIS RIGHT?
U: Yes
S: <answer>
    ...

Paraphrasing a question makes sure the system understood a
query as it was intended, to avoid both a useless costly
search and misinterpretation of the answer by the user.
However, paraphrasing the query and asking the user for
verification each time a question is entered can become
tedious for the user.
```

**Figure 2-10:** Using paraphrase for feedback

## 2.3 Conclusions

Question-answering programs have made valuable improvements over database query languages, as illustrated in this section. Parsers were able to handle a large subset of English syntax, and input constraints were significantly relaxed. Some systems helped the casual user further by providing paraphrasing or guidance in the selection of terms.

There are, however, four serious limitations to the question-answering programs presented so far. First, because systems understand only a subset of English and the extent of that subset is unknown to the user, the user may find it hard to stay within a system's scope of understanding, causing inexplicable inconsistencies to arise. Second, by only allowing the user to pose questions that are readily translated into database queries, these systems implicitly depend on the users' knowledge of the structure and content of the database. Such an assumption is a burden to a user, especially a casual or naive one. Furthermore, because of the limited range of these questions and the fact that the answer was always taken to be the result of a database search, the more general problem of understanding the human question answering process was ignored. Finally, users of an "intelligent" system will tend to attribute human-like features to it, and they will expect the system to respond to their questions in the same way a person would. If systems were able to generate the same kinds of answers a human would provide, interaction would be greatly improved. It is thus important to understand how people communicate and what kinds of answers they expect when asking a question. Understanding the human question answering process is, however, very hard, and researchers have tended to address at most one or two aspects of human interaction in their systems. Some of the results are presented in the next chapter.

## 3. More general approaches to question-answering

The interpretation of a question (or any utterance) goes beyond its syntactic and semantic analysis. This is significant both in understanding a question correctly and in answering it appropriately, even if the question is taken in isolation. See Figure 3-1 for a simple example of this phenomenon.

---

Q: Could you tell me the time?

According to the semantics of this question, it is a Yes/No question, asking for the *capability* of the hearer to inform the speaker of the time. Technically, the answer should be either:

```
                         ''Yes''
or                       ''No'',
meaning:             ''Yes I can''
or                   ''No I  cannot''
```

However, there is no doubt that this question should be interpreted as a *request* to inform the speaker of the time. Thus, a better affirmative response (and the only one acceptable in cooperative human question answering) is:

''It's 2:15 pm''

''No'', on the other hand, is an acceptable negative answer, eventhough it will appear slightly rude. From this answer, assuming the respondent is being at least cooperative (which is a reasonable assumption in everyday communication), we can infer that the respondant does not have a watch and thus cannot inform the speaker of the time. As this answer may appear rude, a better negative answer would be:

''I don't have a watch''

This simple example shows that both the question and the answer are to be interpreted beyond their literal meaning and that, because of the way humans use language, it is possible to make inferences that do not depend on the semantic content of an utterance alone.

**Figure 3-1: Interpretation of a question**

---

Generating a good answer to a question, even in the database domain, can require more than a simple database search. There are several related issues. First, an appropriate answer might have to account for the context of the question, its underlying meaning (questions cannot always be taken literally) and the identity of the user. Second, providing a literally correct answer is usually necessary but it may not be sufficient [36, 62, 31, 29]. Third, it is important to understand what can be inferred from an utterance to ensure that the response provided by the system cannot be misinterpreted by the user. Finally, generating an answer involves both deciding what information to include and how to organize it. All these issues have been studied and the results obtained were applied both to extend the capabilities of natural language interfaces and to form a comprehensive theory of question answering.

Three main approaches have been taken to the problem of understanding questions, determining answers and providing cooperative behavior.

- Categorizing of questions:
    - Lehnert relied mainly on a conceptual categorization of questions to determine their intended meaning and what the answers should be [36].
    - McKeown associated discourse strategies with the question type to determine the content of the answer [45].
- Analyzing a questioner's goals and plans: following Cohen's approach of considering speech acts as actions in a planning systems, Cohen, Perrault, Hobbs, Robinson and Allen looked at answers in terms of how they respond to the questioners' goals. [7, 8, 24, 1]
- Using Grice's cooperative principles [15, 16]: researchers have applied Grice's cooperative principles of communication and the notion of implicature to provide responses that convey the desired information to the user [31, 40, 41, 29].

We will present these approaches in turn and see how they have been applied.

## 3.1 Categorization of questions

### 3.1.1 Providing answers based on the category of the question: QUALM

Lehnert [35, 36] was the first researcher to examine the problem of question answering in the abstract, present a general theory and embody that theory in a system. She tried to identify the factors involved from understanding a question to formulating the answer. Her perspective was one of natural language processing and relied on the idea of conceptual information processing [57] and theories of human memory organization. Determining an answer was based mainly on a conceptual categorization of questions. She showed how a question's context, the state of the system and some rules of conversation could all affect the category of a question and the final answer.

QUALM

Lehnert embodied her theories in QUALM, a system used in conjunction with the story understanding programs PAM (Plan Applier Mechanism) [74] and SAM (Script Applier Mechanism) [9] to answer questions about the stories they had read and to provide a means for evaluating their understanding capabilities.

The process of understanding a question and retrieving the answer was more integrated in QUALM than it was in database front-ends. The input was parsed directly into a language independent representation (Conceptual Dependency notation [56]), and all further processing to correctly interpret the question and answer it occurred at this conceptual level. Scripts, plans and actions were also represented in Conceptual Dependency. QUALM was intended to embody a general theory of question answering and thus tried to answer questions about the stories read in the same way people would. An example of questions answered by QUALM is given in Figure 3-2. In this example, all the answers provided by QUALM include more information than was actually requested in the questions (which are all yes/no questions), making the question answering session more natural and cooperative than if QUALM had always answered "Yes".

The theory of question answering as presented by Lehnert included:

- What it meant to understand a question and how context affected this understanding. Understanding a question in QUALM included classifying it in the appropriate conceptual category. Lehnert defined a taxonomy of questions to predict the kind of memory search that would be required and showed how the category of a question could be derived.

```
Given the story:
```

*John went to New York by bus. On the bus he talked to an old lady. When he left the bus, he thanked the driver. He took the subway to Leone's. On the subway his pocket was picked. He got off the train and entered Leone's. He had lasagna. When the check came, he discovered he could not pay. The management told him he would have to wash dishes. When he left, he caught the bus to New Haven.*

```
In answering a question, QUALM gives more
information than what was actually requested.
Example:

   Q1: Did John go to New York?
    A1: Yes, John went to New York by bus.
   Q2: Did John eat?
    A2: Yes, John ate lasagna.
   Q3: Did someone pick John's pocket?
    A3: Yes, a thief picked John's pocket.

By giving more information than requested in the
question, QUALM provided for a more natural
question-answering session.
```

**Figure 3-2:** A question-answering session with QUALM

- What kind of answers were appropriate. In Lehnert's theory, the kind of answer depended primarily on the question category; once the answer was classified, memory retrieval strategies were determined.

<u>The conceptual category of a question and its derivation</u>

Lehnert claimed that her conceptual categorization was crucial in determining the answer to a question because it indicated what a question really meant, i.e., what the intentions of the questioner were. An incorrectly classified question would yield an inadequate answer. The thirteen conceptual categories she identified are shown in Figure 3-3. In the theory presented by Lehnert, a question was parsed on several levels in order to be correctly classified. First, a lexical processing level produced the literal semantic interpretation of the question. Second, a reference recognition level identified what the question referred to in memory. At this point, the question had been understood in a literal way and had been parsed into a Conceptual Dependency representation. A discrimination net then tested the structure of the question to place it into one of the conceptual categories. The classification obtained by going through the net was not necessarily the final one, since it relied only on the structural features of a question, and thus on its literal meaning. Finally, inferences based on dialog conventions, context, and knowledge of the user were employed to refine the categorization of the question and add constraints if necessary. Refining the categorization corresponded to transforming the *literal* meaning of a question into its *intended* meaning. Constraints were also sometimes used to restrict the set of possible answers. Lehnert maintained that without such constraints,

```
Causal Antecedent: e.g. ''Why did the book fall?''
            [the question asks about the event causing a concept]

Goal Orientation: e.g. ''Why did Mary hit John?''
            [the question asks about the mental state of the actor]

Enablement: e.g. ''How was John able to sleep?''

Causal Consequent: e.g. ''What did John do after quitting his job?''

Verification: e.g. ''Did you eat?''

Disjunctive: e.g. ''Is Data Structures on mondays or tuesdays?''

Instrumental/procedural: e.g. ''How  did  you  get to  school?''
                                 (by bus)

Concept Completion: e.g. ''What did you eat?''

Expectational: e.g. ''Why didn't you come last night?''

Judgmental: e.g ''What should I do now?''

Quantification: e.g ''How many Pascal books do you have?''

Feature Specification: e.g. ''How old is your dog?''

Request:  e.g. ''Do you have the time?'' (3 am)
```

**Figure 3-3:** Conceptual categories identified by Lehnert

answers that are correct but useless and inappropriate would be generated[4].

Lehnert divided the inference mechanism into three parts: the contextual-inference rules, the context-independent rules and the knowledge-state assessment rules. Each rule changed the category associated with a question or

[4]Example of a constraint proposed by Lehnert to further specify a question:

*Universal-Set inference rule:*

*Use the active script to derive the set of potential answers.*
*Add the constraint that the answer must belong to this set.*

Consider the question:

*Q: Who was not in class today?*

This rule is needed to avoid producing the following answer:

*A: Louis XIV, Napoleon and Charlemagne.*

Such an answer would be inadequate (though technically correct, even though it is a partial answer). According to Lehnert, this problem would arise because no constraint on the set of possible answers was added (i.e. that the answer should be from the set of students for that class).

added a constraint to the question specification. A rule had preconditions that had to be true in order for it to apply and a target interpretation indicating the change to be made.

**Contextual-inference rules** examined the conversational context of a question and allowed for a question to be understood within that context. The context of the question in QUALM could be provided either by the *topic* of conversation or by a *script*.

In human communication, the topic of a conversation is heavily used to understand utterances, for it provides a context in which sentences can be rendered unambiguous, and ellipsis and anaphora understood. Keeping track of the topic of the conversation is essential. As an example, when one knows the conversation topic and assumes that there is continuity in the conversation, it is possible to understand the ellipsis in the dialog (1) shown in Figure 3-4. The topic of conversation together with a transcript of past discourse can be potentially more useful than the ellipsis and anaphora resolution we saw in the natural language front-ends to databases, where ellipsis was solved by matching the current utterance against the previous one and anaphora resolved by finding the closest matching element[5]. The topic of conversation can indeed provide a more complete context for a question. However, in QUALM, the conversation topic as a whole is taken to be the *last* conversation topic. As a result, the ellipsis and anaphora resolution procedure used is essentially the same as the ones previously studied.

```
1) Topic of conversation

P1: Have you seen Mary recently?
P2: She is out of town visiting her mother and her sister.
P1: When is she coming back?

In this  dialog, the  topic  of conversation  is  ''Mary''
Using traditional pronoun resolution, ''she'' in the  last
sentence would be believed to refer to either  ''sister'',
''mother'' or ''Mary''.  The participants, however, have
no  trouble  identifying  the  pronoun  as  referring  to
''Mary''.

2) Knowledge about scripts

Within the script of an auction, the question:
      ''Who will give me $50?''
means: ''Who will bid $50?''

3) Beliefs about the questioner

From a father to his daughter coming back at 2:00 AM.
Q: Do you know what time it is?
Being aware that her father surely knows the time, she should
interpret the question as ''Why are you coming so late?''
```

**Figure 3-4:** Factors influencing the interpretation of a question

Some stereotyped situations are associated with specific conversational style. Such a situation, called a *script*, can

[5]See [18] for more on this topic.

be the context for a question; within a particular script, a question may have a very specific meaning, which differs considerably from its usual meaning, as illustrated in the second example in Figure 3-4. In her theory, Lehnert suggested that a program should use knowledge about scriptal conversational patterns to help understanding a question.

**Context-independent inferences rules** transformed the interpretation of a question from the literal one to the intended one, causing the new interpretation to reflect the way people use language. Figure 3-5 shows one such rule, used to transform yes/no (*verification*) questions into *requests*.

---

```
Agent-Request conversion rule:

    Criteria:   1. Conceptual categorization = Verification

                 2. Question concept is of the form:

                     an object X is in Z's possession;
                     the modal of the question is CAN[6].

    Target interpretation:
                    Z is to give X to Y, where Y is the person
                    posing the question.

Using this rule, the question:

                Can I have a cookie?

is re-interpreted as:

                Would you give me a cookie?

This rule embodied the fact that a restricted set  of
yes/no questions should  be interpreted as  requests.
The application of this rule, as specified in  QUALM,
does not depend on the context of the question.
```

**Figure 3-5:** A context-independent rule in QUALM

---

**Knowledge-state assessment rules** involved beliefs of the answerer about the questioner. In Lehnert's theory, such knowledge mainly affects how a question is interpreted and thus how it is classified, as opposed to how it is answered. As an example, a question can acquire an interpretation different from its usual one if it is obvious that the questioner already knows the answer to the usual interpretation (see (3) in Figure 3-4). While Lehnert gave some heuristics for how such knowledge could affect the interpretation of the question, she did not address the problem of representing that knowledge, and the rules were not implemented.

Answering a question

[6]The rule, as specified by Lehnert actually uses Conceptual Dependency notation for the question form. The form showed in this example is one of two forms described in the *criteria* for this rule.

To answer the question, a system must first decide what to include in the response. In QUALM, this was done mainly on the basis of the question category. Lehnert recognized that humans have the ability to vary their answers but avoided studying all the factors that can influence an answer. She chose to vary the style, or *mood*, of the system instead. The mood here referred to whether the system was to provide a "talkative" answer or not, as shown in Figure 3-6.

```
Q: Did John go to New York?

Mood = talkative
        A1: Yes, John went to New York by bus.


Mood = minimally responsive
        A2: Yes.
```

**Figure 3-6:** How QUALM could vary an answer by varying its mood

Based on mood level, QUALM varied the *amount* of information given in the answer using the heuristics (*elaboration options*) specified by Lehnert. Of these heuristics, few were actually implemented in QUALM, and the remainder were too vague to be applied at this point, since they require a deeper understanding of how knowledge of the user affects an answer. Moreover, this scheme allows the answer to be varied only in the amount of information given to the user. If the system was to answer the same way a person does, the answer should vary both in the *amount* and the *kind* of information given. It may indeed be desirable to give different kind of information depending on the point of view taken to answer the question [46], the type of the user [53], or the user's knowledge of the domain [48]. QUALM could vary its mood only when the appropriate variable is changed by hand. Ideally, this tailoring should be done dynamically.

Once it was decided what information to include in the answer, a memory search depending on the conceptual category of the question was carried out[7]. Lehnert noted that questions about events that did not happen (why-not? questions) were more difficult to answer than other kinds of questions. QUALM was able to answer such questions so long as they were about failed expectations, that is about events which we would expect to happen, because they were embodied in a script, but did not occur (also called "ghost paths"). To do so, the program used its script knowledge to detect where the departure from the script occurred.

Finally, Lehnert studied the problem of choosing the best answer when several are possible. She defined rules that should be followed in picking the best answer. However, those rules involved knowledge about the user and were thus not yet ready to be used in a question answering program without further study.

## Conclusions

Lehnert's departure from the database domain was important as it provided a more general framework for question-

[7]The Request category was not studied in QUALM. Because a request question typically asks for an *action* to be performed, and asking for the performance of an action (as in "tell me the date") is not appropriate in the story understanding context, Lehnert argued that there was no need to study request questions. However, consider the question "Can you tell me something about John", which is intended to mean "Tell me what you know about John, what did he do?". Such a question would require the program to generate a summary of the story.

answering research. Through the categorization of questions and the rules used to transform a question from the literal to the "intended" interpretation, Lehnert provided a fairly simple mechanism to answer questions in an appropriate manner in many cases and successfully implemented it in QUALM to answer questions about stories. However, deriving the proper question classification will not always lead to an appropriate answer. A simple example is shown in Figure 3-7, where, based on the category of the question (verification), the answer would be *"Yes"* or *"No"*, or, using an elaboration option, a statement specifying the immediate cause. These answers would not be as informative as the indirect answer given in the example, which is by far more appropriate and cooperative.

```
Q: Are you ready for your exam tomorrow?

This question is a verification question.  It  could
be  answered  with  the  direct  answers  ''yes''  or
''no''.  Lehnert specified an elaboration option (the
enquiry/explanation option) which  could generate  an
indirect answer by finding an apparent cause.   Here,
an apparent cause could be:

A1: I have not studied.

The following answer, however, is more appropriate
and informative in this case:

A: I decided against taking it.
```

**Figure 3-7:** An answer not dictated by the category of the question

Furthermore, while Lehnert was able to identify several of the factors necessary to understand a question and determine its answer, the rules presented in the theory do not constitute a general enough theory of language usage to fully explain how humans interpret questions and make appropriate answers. Important issues in the question-answering process such as discourse structure and knowledge about the user and what makes an answer appropriate remains to be studied in depth.

### 3.1.2 Providing coherent answers using discourse structures: the TEXT System

Like Lehnert, McKeown [45, 44] used the categorization of questions to aid in determining the answer to a question. Her research, however, was not aimed at understanding questions but rather at determining the content and textual shape of answers. She studied how principles of discourse structure and focus constraints could be used to guide the information retrieval process.

#### The TEXT system

McKeown's work was implemented in the TEXT system. The TEXT system used a portion of an Office of Naval Research database containing information about vehicles and destructive devices. The knowledge base contained entities, relations between them, attributes, a generalization hierarchy, and a topic hierarchy. Using the TEXT system, a user could ask questions such as those in Figure 3-8. Since the answers to these questions are not the result of a database search, deciding what to say becomes an important issue. McKeown developed means to select the appropriate information from the database and organize it coherently into paragraph-long answers. Though developed mainly to study the problems of natural language generation, the TEXT system is a useful tool to add to a conventional database interface, since it allows users to familiarize themselves with the database content.

```
1. What is a frigate?
2. What do you know about submarines?
3. What is the difference between a whisky and a kitty hawk?

The TEXT system could handle questions about the database
content.
```

**Figure 3-8:** Questions answered by the TEXT system

Schemas

Instead of just tracing through the knowledge representation to select the answer, McKeown employed rhetorical techniques found in naturally occurring texts. *Rhetorical predicates* are among the means available to speakers to effectively convey information. They include: *analogy*, specification of a property or *attribute* of an object (entity), and *illustration by example*.

Linguistic studies showed that certain combinations of these predicates were preferred over others [17]. Furthermore, after studying English texts and transcripts, McKeown found that certain combinations were associated with particular discourse purposes. For example, the definition of an object was often accomplished using the following sequence of rhetorical predicates:

(1) Identification of the object as a member of some generic class;
(2) Description of the object's function and attributes;
(3) Analogy to familiar objects;
(4) Illustration by examples.

McKeown encoded these combinations of predicates into *schemas*, which listed the different predicates that could be used for a particular discourse purpose in the order in which they should appear. Thus, schemas represent standard patterns of discourse structure. The TEXT system used four schemas, each corresponding to one or more discourse purposes (see Figure 3-9).

| Schema | Discourse purpose |
|---|---|
| Identification: | requests for definitions |
| Attributive: | requests for available information |
| Constituency: | requests for definitions<br>requests for available information |
| Compare and contrast: | requests about the differences between objects |

**Figure 3-9:** Schemas used in the TEXT system

Figure 3-10 shows the *identification* schema. The question types in TEXT correspond to different discourse purposes. Consider the questions in Figure 3-8. The first question is a request for a definition, the second a request for information about an object, and the last asks for the difference between two objects.

```
                    Identification Schema[8]

    Identification (class & attributive / function)
    [Analogy/ Constituency/ Attributive/ Renaming]*
    Particular Illustration / Evidence +
    [Amplification  / Attributive / Analogy]
    Particular Illustration / Evidence
```

**Figure 3-10:** Example of a schema used in TEXT

Selecting the answer from the knowledge base

When TEXT was asked a question, it selected a subset of the knowledge base that contained all the relevant information to that question. This step limited the amount of information future processes had to look at in order to pick out the appropriate answer. This subset of the knowledge base was called the *relevant knowledge pool* and was constructed based on the question type. For instance, for a request about definitions, all the information immediately associated with the object being defined (such as its attributes, superordinates and subtypes) was selected.

A schema was then chosen based on the discourse purpose associated with the question type and the amount of information available in the relevant knowledge pool. The elements of the schema were filled sequentially, and its predicates were matched against the relevant knowledge pool. An instantiated predicate corresponded to a proposition that would be translated into an English sentence by the generation component [44]. When alternate instantiations were possible, the system used constraints on how focus of attention can shift to select the most appropriate one. McKeown extended previous work on focus of attention (for understanding anaphora) [64] to provide focus constraints that limit what can be said next and still maintain coherence. Using these constraints, the program avoided choosing propositions that jumped from one topic to another in an uncontrolled manner, thus ensuring the text coherence. Essentially, the focus guidelines caused the program to:

- Select the proposition such that the focus changes to an item just introduced.
- Select the proposition that allows the focus to remain the same.
- Choose the proposition that makes the focus switch back to the previous one.

Using the identification schema and focus constraints, TEXT produced the answer shown in Figure 3-11: Note that, based on the knowledge base and focus constraints, only five predicates were chosen from the schema.

Conclusions

McKeown concentrated on strategies for constructing an answer based on the discourse purpose sought after. Using discourse strategies (that appear to exist in natural language texts), and focus constraints, the TEXT system was able to produce coherent paragraph-long answers. However the user's goal in asking the question was considered only insofar as the question type characterized this goal. How the user's needs, beliefs, and knowledge can influence the final answer was left unstudied. Finally, the TEXT system did not attempt to vary its answer depending on the user.

[8]We are using McKeown's notation: brackets ("[ ]") indicate optionality, "/" alternatives, "+" that the item may appear one or more times, and "*" that the item may appear zero or more times.

```
(definition AIRCRAFT-CARRIER)

Schema selected:  identification

predicates instantiated:  (1) identification
                          (2) analogy
                          (3) particular-illustration
                          (4) amplification
                          (5) evidence

1) An aircraft carrier  is a surface  ship with a  DISPLACEMENT
between 78000 and 80800 and a LENGTH between 1039 and 1063.  2)
Aircraft carriers have  a greater LENGTH  than all other  ships
and a  greater DISPLACEMENT  than most  other ships.   3)  Mine
warfare ships, for example,  have a DISPLACEMENT  of 320 and  a
length of 144.  4)  All aircraft carriers  in the ONR  database
have REMARKS of  0, FUEL TYPE  of BNKR, FLAG  of BLBL, BEAM  of
252,   ENDURANCE-RANGE   of   4000,   ECONOMIC-SPEED   of   12,
ENDURANCE-SPEED of 30, and PROPULSION  of STMTUGRD.  5) A  ship
is classified  as  an  aircraft carrier  if  the  characters  1
through 2 of its HULL-NO are CV.
```

**Figure 3-11:** Example of an answer generated by the TEXT system

The system could be improved by incorporating a user model to add constraints on which predicates are to be instantiated and decide on whether recursive use of the schemas is appropriate.

### 3.1.3 Summary

Examining questions other than those that can be readily translated into a database query, researchers started to look at the problems of interpreting a question correctly (i.e. as it was *intended* to be interpreted) and answering it appropriately (i.e. providing an answer that satisfies the questioner). Clearly, systems with such capabilities would provide a more graceful interaction to the users. Lehnert saw the problem of question-answering as one of placing questions in appropriate categories and concentrated on *understanding* questions, using primarily the question's context and some conversational rules. McKeown, on the other hand, concentrated on providing an informative and coherent answer, based on discourse structure and focus rules found in natural language texts. In her system, McKeown assumed that the user's purpose for discourse was known and did not study parsing issues. Making use of a question's context, discourse strategies and focus constraints in providing an answer definitely results in a "better"[9] answer than one resulting from a simple database retrieval. However, neither Lehnert nor McKeown characterized what an *appropriate* answer really is and this issue remains to be studied. Finally, in both systems, the problems of generating indirect answers when necessary and tailoring an answer to the user were neglected.

[9]"Better" here refers to how natural and close to a human response the answer is.

## 3.2 Appropriate responses and the goal of the questioner

Another approach to the problem of determining an answer was proposed by Hobbs/Robinson and Allen/Perrault [24, 1][10]. Their *goal-oriented* approach attempted to explain indirect answers and cooperative behavior in terms of the goals of the questioner.

The problem in providing appropriate answers lies in the fact that it is very hard to pinpoint what *appropriate* means. By not addressing the question itself but still providing enough information for the user to infer the answer, indirect answers provide a good basis for studying the appropriateness of answers. Furthermore, in human question answering, indirect answers are often given. We can probably assume that, when engaged in cooperative dialogues, if humans choose to answer a question indirectly, the indirect answer is more appropriate then a direct one. It is therefore desirable for a system to also be able to provide indirect answers when necessary. While people have no problem understanding such answers and generating them, it is hard to characterize what makes an indirect answer a good response and when one is preferable to a direct one.

Hobbs and Robinson studied indirect responses in task oriented dialogues with the goal of determining what makes an answer appropriate. After studying transcripts of task oriented dialogues, they divided indirect answers in three categories:

1. The answer was indirect but did answer the question asked; the direct answer could be inferred from it.
2. The answer denied a *presupposition*, a belief the questioner had about the domain[11].
3. The response addressed higher goals the questioner was trying to achieve.

An example for each of these cases is presented in Figure 3-12.

Assuming that people ask questions with goals in mind, Hobbs and Robinson concluded from their analyses that, to be appropriate, an answer must provide some information that allows questioners to achieve their goal. That information need not be exactly the same as that asked for in the question. The answerer may find the question inappropriate given the goal to be achieved and provide information that he thinks is relevant.[12] This sort of behavior assumes the answerer (1) wants to help the questioner, (2) is able to help the questioner and (3) knows the questioner's goal.

Hobbs and Robinson thus proposed a characterization of the appropriateness of an answer. However, the guidelines they offer do not yet specify how to generate an "appropriate" answer, nor do they determine when an indirect answer should be chosen over a direct one. Finally, it is a significant problem to figure out the goal a person wants to achieve.[13] This is complicated by the fact that one utterance may be used to achieve several purposes.[14]

[10]The results obtained by Allen and Perrault will be presented in the chapter discussing user modelling problems, for they were mainly concerned with the problems of inferring the goals and plans of the user, which involves forming a model of the user.

[11]Special cases of this problem have been studied at length, as will be seen in the next section [31, 40, 41].

[12]These issues have also been studied by Joshi [29] who tried to identify the kinds of responses expected in certain situations (including that of an expert and a novice) to make sure that, because of the expectations from the questioner, the answer will not be interpreted in an unintended way.

[13]This problem was addressed in part by [1].

[14]This has been termed the *multifaceted aspect* of utterances [19].

1) Although indirect, the answer answers the question asked.

Q: Is your area paper finished?
R: I just have to add the bibliography.

The direct answer (NO) can be inferred from the response given. However, the indirect answer provides more information than the direct one would have had.

2) The answer denies a presupposition of the question.

Q: Have you passed your oral exam?
R: We don't have an oral exam in our department.

The questioner had the assumption that the hearer had to take an oral exam. The respondent corrects that belief.

3) The response answers to higher goals the questioner was trying to achieve.

Q: Which key do I need?
A: The door is unlocked.

The respondent recognizes that the questioner's goal is to open the door. The response he provides does not answer the question but still enables the questioner to fulfill his goal.

**Figure 3-12:** Examples of indirect answers

## 3.3 Grice's cooperative principles and the notion of implicature

Instead of deriving answers based on question taxonomies or the goal of the questioner, Kaplan [31], Mays [40], McCoy [41], and Joshi [29] attempted to use the more general theory of *language usage* as developed in the area of pragmatics in order to characterize what an appropriate answer is.

Pragmatics, the study of how knowledge about the context of an utterance affects its understanding, gives us a theory to explain how people can mean more than what they say with the notion of *implicature* [70, 37]). Studying how implicature works provides a general way to deal with the issues of language usage, instead of applying the specific rules used by QUALM. It is, however, a very hard problem and has been studied only in part.

### Grice's maxims and implicature

Grice [15, 16] proposed a theory about how people use language. He suggested that the following guidelines are used in human conversation in order to communicate efficiently and in a cooperative manner:

1. **The Cooperative Principle.** Make your contribution as informative as is required, in the context in which it occurs.
2. **The Maxim of Quality.** Do not say what you believe is false or what you lack evidence for.

3. **The Maxim of Quantity.** Make your contribution as informative as necessary, but not more informative than required.
4. **The Maxim of Relevance.** Make your contribution relevant to the conversation.
5. **The Maxim of Manner.** Be brief and orderly. Avoid ambiguity and obscurity.

Grice claimed that, when engaged in a conversation, humans tend to cooperate with each other and will use the guidelines given above. It follows that, even when an utterance seems to violate the principle of cooperation, humans will try to interpret it at a deeper level to make it in fact cooperative. This interpretation will give rise to inferences that could not have been drawn from semantics only. This framework can be used to understand indirect responses, as shown in the example in Figure 3-13. The inferences that can be drawn from our knowledge of language usage are called *conversational implicatures*. Understanding these conversational implicatures would clearly help in building computer systems that would be able to understand questions more deeply and provide indirect answers.

There are two ways implicatures can arise: by following Grice's maxims or by apparently flouting them (that is deliberately violating them as in the second example of Figure 3-13).

---

```
1) Following Grice's maxims:

  Q: How cold is it outside?
  A: I am only wearing a sweater over my T-shirt.

The answer does not seem to be relevant. However, we do
consider it to be an appropriate answer to the question posed.
Assuming the answerer is being cooperative, we try to find a
connection between the person's outfit and the outside
temperature to derive the direct answer.


2)Flouting Grice's maxim (The maxim of quantity this case):

     Q: What was the party?
     A: A party is a party.

This statement is rather obvious and thus violates the maxim
of quality.  However, it does convey information, namely
that that the party was average and not overly exciting;
e.g, there is nothing else to say.
```

**Figure 3-13:** How inferences can be drawn from an answer

---

The information conveyed both when following and flouting a maxim is part of a comprehensive theory of language usage. However, to implement question answering programs that provide a natural access to some body of data, we are more concerned with utterances that follow Grice's maxims.

Grice's maxims and the notion of implicature have been used in an attempt to characterize when indirect answers are preferable over direct ones. In particular, it was found that indirect answers may be preferable to direct ones when the latter are negative, for then, indirect answers may be more informative. Two examples are shown in Figure 3-14. Two cases of this situation have been studied: presumptions, and scalar implicature. We first

Q1: ''Are all the professors in the Linguistics department full professors?''

A1, a direct answer: ''Yes''
A2, an indirect answer: ''There is no Linguistics department here''

A2 is more informative than A1, even though A1 is also (trivially) correct. Moreover, A1 can actually be misleading, by not refuting the belief that there is a Linguistics Department.

---

Q2: ''Are all the professors in the Linguistics department full professors?''

A1, a direct answer: ''No''
A2, an indirect answer: ''two are''

Here again, the indirect answer is more informative than the direct one.

In both examples, by providing A1, a system would be considered to provide only partial information. (This is called *stone-walling*.)

**Figure 3-14: Indirect answers may be more informative than direct ones**

introduce these two cases and then present how they have been studied in order to improve question answering systems.

Presumptions

Consider the question:

Q3: What grade did Bob get in Data Structures?

In asking this question, the speaker must believe that Bob took the Data Structures course. In fact, this follows from one of the conventions of cooperative conversation: the questioner must leave a choice of direct answers to the respondent, or, in other words, the questioner must believe that there are several possible direct answers to his questions. (If there was at most one possible direct answer, he could then infer the answer without asking the question.) So, if all but (at most) one of the direct answers entails a proposition *P*[15] , we can assume that the questioner believes *P*. Proposition *P* is then said to be *presumed* by the question. In the previous example, Q3 *presumes* that "Bob took a Data Structures course" and "there are more than one possible grade".

If the presumption is false, the answerer should refute it, and an indirect answer is more appropriate than a direct one. Failure to deny the presumption implicitly confirms it and thus misleads the questioner. Such a situation arises

[15]A proposition *P1* semantically entails the proposition *P2* if and only if *P2* is true whenever *P1* is.

when (1) a question contains some of the speaker's beliefs about a domain, that is the question *presumes*[16] some fact to be true about the domain of discourse [33, 37]; and (2) these beliefs are not supported by the data (or facts). The user has then a misconception about the domain. Because of a natural language system's apparent understanding capabilities, users are prone to expect the system to correct any misconceptions they may have. For a system to be cooperative, it must be able to detect such misconceptions and answer in such a way as to correct the misconception and avoid misleading the users. In order to detect whether the questioner has some misconceptions about the domain of discourse, a system needs to be able to infer the presumptions of a question. A false presumption corresponds to misconception on the part of the user. Kaplan [30] called a question containing a false presumption a *loaded* question. He was concerned with detecting misconceptions and providing appropriate answers to loaded questions.

Scalar implicature

The second example in Figure 3-14 involves a particular type of implicature, called *scalar implicature*. Horn [25, 26] observed that when an utterance refers to a value on some scale defined by semantic entailment, that value represents the highest value on the scale of which the speaker can truthfully speak. Assuming the speaker follows Grice's principles, we can then infer that he is saying as much as he can. Consequently, higher values on the scale are either false or unknown to be true by the speaker, and values lower on the scale can be marked as true, since they are entailed (see Figure 3-15). Horn called this phenomenon *scalar predication*, while Gazdar [13] called it *scalar quantity implicature*. As illustrated by the example, a system that can deal with such implicatures would be more cooperative by being able to provide indirect responses.

```
Consider semantic scale <all some none>
and the exchange:

Q: Did you buy  the books required for the qualifying exam?
A: I bought some.

The direct answer ''No'' can be inferred, since, by the maxim
of quantity, if ''all'' the  books were bought, the  answerer
would have  said so.  The scalar  implicature allows  us  to
interpret the answer as:

'' There are some books that I have not bought.  However, I
did buy a subset of them.''

Here again, the  indirect answer is  more informative  than
the direct one.
```

**Figure 3-15:** Example of scalar implicature

Grice's cooperative principles and the notion of implicature were used to extend the capabilities of natural language interfaces to database systems and make them behave in a more cooperative and human-like manner.

[16]A more specific case happens when all the direct answers to a question entail a proposition *P*, in which case the question is said to *presuppose P*.

### 3.3.1 Extensional misconceptions and the CO-OP system

The ability to correct a misconception a user may have about a domain can be very important for a natural language interface that is to be used by naive and casual users who are often not familiar with the structure or the content of the domain and are therefore likely to have false beliefs about it. Kaplan [31] was the first researcher to address the problem of correcting a misconception on the part of the user.

Kaplan studied questions that could be translated into a single database query and was concerned with those containing *extensional presumptions* (i.e, presumptions about the existence of a set of entities in the knowledge base) (see Figure 3-16). Kaplan used language-driven inferences (those that can be inferred from *language usage*) as opposed to domain-driven inferences (those that need domain specific knowledge as well as general world knowledge) to detect presumptions in questions. In the CO-OP system, he showed that, by limiting the domain of discourse to database queries, a significant class of presumptions could be computed using only language-driven inferences.

The computation of the presumptions of a database query was possible by representing the query in an intermediate graph notation, the *Meta Query Language* (MQL). A database query can be viewed as requesting the selection of a subset from a presented set of entities [2] by putting this presented set through a series of restrictions. In CO-OP, this process was done by translating the query into the Meta Query Language. The nodes of the graph were *sets* and the arcs *binary relations* among them. Using this scheme, the direct answer to the query was obtained by composing the sets according to binary relations, in fact selecting a subset (see Figure 3-16). The importance of this representation for detecting misconceptions lies in the fact that each connected subgraph of this representation corresponds to an extensional presumption of the query (the existence of a set), as illustrated in Figure 3-16. Consequently, an empty subset indicates a false presumption, and hence a misconception on the part of the user.

CO-OP detected misconceptions while retrieving answers from the database. If the query resulted in an empty set, the intermediate representation was examined. Each connected subset was checked for emptiness in turn, and the corresponding corrective answer was generated:

```
Example of a corrective answer:

''I don't know of any Linguistics courses.''
```

By translating a query into the MQL intermediate representation, Kaplan was able to compute presumptions in a domain-independent manner and without the need to add general knowledge to the knowledge base. The generation of the corrective answer was also done in a straightforward manner, by checking for the possible emptiness of the sets. Even though CO-OP was able to offer corrective responses only to questions showing extensional misconceptions, this work was important as the starting point for studying how to detect misconceptions in questions and how to respond appropriately.

### 3.3.2 Intensional misconceptions

The misconceptions dealt with in CO-OP are those that depend on the *content* of the database. They arise from *extensional* presuppositions, meaning that the user believes that there are elements satisfying a particular set description when that description actually has no extension in the database. Another type of misconception, called *intentional* misconceptions, depends on the *structure* of the database. It occurs when the user believes that an entity (or a set of entities) can participate in a relation when, in fact, it cannot. These misconceptions are not addressed in CO-OP. An example is shown in Figure 3-17.

The query:

  Which students got F's in Linguistics courses?

was represented as follows in Meta Query Language:

```
  ------------                    ----------
/ students    \      GOT        /    F's    \
\             / -------------> \            /
  ------------                    ----------
                                      | IN
                                     \|/
                                      |
  ------------                    ----------
/linguistics  \                 / courses   \
\             / -------------> \            /
  ------------                    ----------
```

This query presumes the existence fo the following *sets*:

     There are students.
     There are F's. (Some of the grades are F's.)
     There are courses.
     There is Linguistics.
     There are students that got F's.
     There are grades of F's in (some) courses.
     There are Linguistics  courses.
     There are students who got F's in courses.
     There are students who got F's in Linguistics courses.

**Figure 3-16:** A query in CO-OP

Q: Which undergraduates teach courses?

Q presupposes that undergraduates *can* teach courses. If in fact, undergraduates cannot teach courses, the following answer is not appropriate for it does not deny the presupposition.

A1: I don't know of any undergraduates that teach courses.

A better answer would be:

A2: Undergraduates cannot teach courses.

If undergraduates *could* teach courses but were not teaching this particular semester, the misconception would be *extensional*.

**Figure 3-17:** An intensional misconception

In order to recognize such misconceptions, the system needs some knowledge about the permissible relations between entities in the knowledge base. Mays [40] designed a data model aimed at allowing a system to detect intensional misconceptions. This data model included the following information:

- entity-relation information: what entities (or group of entities) can participate in what relations.
- hierarchical information: superset/subset information.
- partition information: the incompatibility of groups of entities.

Using this database model, an intensional misconception is recognized when a relationship between entities presupposed in the query cannot be established in the database model. This failure can be computed in a fashion similar to that used in Kaplan's system. Here, the MQL query is checked against the database model (instead of against the database) to verify that each relation presupposed can be established.

Mays concentrated on *detecting* intensional misconceptions. The implementation of the component of the system that would actually respond to a question showing some misconceptions on the part of the user was not carried out.

### 3.3.3 Current work on misconceptions

Yet another type of misconception happens when there is a discrepancy between what the user believes about an object and what the system believes about it. McCoy [41] is currently investigating these *object related misconceptions*, mainly in the framework of expert systems. She is concerned with how to *correct* the misconception as opposed to detecting it (which was the emphasis of both Kaplan and Mays). McCoy is examining the problem of characterizing in a domain independent manner what influences the choice of additional information to include in answers and enabling a system to produce such responses.

Instead of relying on an *a priori* list of possible misconceptions, as in the approach taken in some CAI systems [68, 3, 67], McCoy classified object related misconceptions based on the knowledge base *feature* they involve. A feature of the knowledge base could be a *superordinate* relation or an *attribute*. Through the studies of transcripts, she has identified what types of additional information should be contained in the answer corresponding to each type of object related misconceptions. A correction schema that dictates what kind of information to include in the answer is associated with each type of misconception.

### 3.3.4 Using implicature to provide indirect answers to yes/no questions

Through the work on misconceptions, we have seen how indirect answers can be provided to yes/no questions, by computing the presumptions contained in the questions and refuting them if necessary. Besides being more informative, these indirect answers can also avoid misleading the user. Traditionally, the "normal" answer to Yes/No questions is chosen among the two direct answers "Yes" and "No", and systems that allow for indirect answers go through extra processing to recognize that an indirect answer is necessary and to derive it. The notion of implicature, however, provides a way to broaden the set of possible responses to a Yes/No question.

Hirschberg [23] proposes a redefinition of yes/no questions in order to treat indirect answers in the same way as direct answers, i.e. as a possible response to yes/no questions. She uses the notion of *scalar implicature*, and extends the definition of *scale* to include *set/set-member*, *whole/part*, *process stages*, *spatial relationship*, *prerequisite orderings*, *entity/attribute*, *isa-hierarchy*, and *temporal* scale. By broadening the definition of scale, the direct answer to the question below can be entailed using scalar implicature:

```
Q: Have you baked the cake?
A: I just put it in the oven.

With the process stages scale: <mix; put in oven; bake; frost>
the hearer can infer  that the values higher  up on the scale  are
either false or unknown to the speaker.  The direct answer  ''No''
can  be  entailed,  but   indirect  answer  given  provides   more
information.
Note that this scale could be embodied  into a script.
```

If the set of possible answers to the questions contained all the values in the process stage scale (instead of only the two direct answers "yes" or "no"), the system would be readily able to provide the more informative indirect answer based on the information contained in its knowledge base.

Based on this extension of scalar implicature, Hirschberg thus proposes a new way to represent Yes/No questions to broaden the set of possible answers. A system that can use scalar implicature will be able to provide more useful answers to Yes/No questions and avoid conveying false inferences that may be drawn by direct answers.

Unfortunately, scalar implicatures do not explain all the possible indirect answers to Yes/No questions however. In particular, they do not support request type questions. Moreover, it may be very hard to determine all the scales that are applicable at any point and which scale to use.

## 3.3.5 Avoiding false inferences

Another interesting issue related to conversational implicatures is that of false inferences, that is inferences that could be drawn from an utterance but that the speaker knows are false. In cooperative behavior, it is clearly undesirable to produce such an utterance. Likewise, a system should not provide an answer from which the user could draw false inferences. Again, such an answer would be considered misleading.

To study this situation, Joshi [28] modified slightly the maxim of quality which became:
*"if you, the speaker, plan to say anything which may imply for the hearer something that you believe to be false, then provide further information to block it"*)
Using this modification, Joshi, Webber and Weischedel [29] argue that, to be cooperative, a system must be able to recognize that a response may mislead the user and modify such a response. They are attempting to characterize the cases in which the system can foresee the possibility of drawing wrong inferences from the answer. To do so, they first develop a formal method for computing the possible inferences that can be drawn from an answer, identifying the factors that come into play, and characterize the types of behavior usually expected from the answerer.

They identified the types of informative behavior in cases where the question indicates that the questioner wants to achieve some goal. They distinguished between the *stated goal* or *S-goal* of the user (that is the goal as stated in the question) and the *intended goal* or *I-goal*, which represents what the user really wants to achieve. They identify relations that can exist between these two goals. As an example of such relation, the I-goal may be an *enabling* condition for the S-goal. Assuming a system knows the actions and events that can achieve a goal, together with their probability of occurring, the system can detect whether, by achieving the stated goal, the intended goal will also be achieved. The system can then generate an answer, based on the relation between the two goals and the information contained in the database.

Joshi, Webber and Weischedel characterize the information to be given in a response according to whether achieving the S-goal will also achieve the I-goal. A possible situation is given in the example below:

<u>Example</u>:

```
    The proposed action X does not achieve the I-goal: G

    Q: I need to fulfill my elective requirements.
       Can I take Graph Theory?

  If Graph Theory does not fulfill the elective requirement
  (i.e, the action does not achieve the goal),
  a  cooperative answer might be:

    A: You can take Graph Theory, but it will not
       fulfill your elective requirements, as
       electives need to be humanity courses

  In this  case,  the  system  would check  that  G  cannot  be
  achieved by performing X, even though the enabling conditions
  of X are true, so that X can be performed. On the other hand,
  the system finds that the action Y can achieve the goal,  and
  thus informs the user.  It  would be misleading the user  not
  to inform him that performing X does not achieve the intended
  goal.
```

Using this method, a system would be able to inform users of the effectiveness of their action in order to achieve their goal, and provide alternatives if necessary. However, as noted before, inferring both the stated goal and the intended goal is a very hard task. Furthermore, the process of generating informative responses is static (conditions are checked in turn, as in a discrimination net) and there is no explicit reasoning about the questioner's expectations.

## 3.4 Conclusions

Progressing beyond the study of database front-ends allowed researchers to address the problem of question-answering in a more general framework. Through the study of language usage, theories about human question-answering were developed, and methods to alleviate the problems of previous question-answering programs were implemented. Researchers analyzed more thoroughly the factors involved in interpreting a question and found that the context of questions together with some rules of conversation could help understanding a question beyond its literal meaning. Using these results, they were able to build programs that attempted to extract the *intended* meaning of questions. Other researchers tried to characterize the notion of *appropriateness of an answer*, by looking at the goals of the users and the rules of cooperative discourse. More graceful man-machine interaction resulted as programs were able to answer in a more human-like manner, providing indirect answers, correcting misconceptions on the part of the user, and answering questions about a knowledge base.

# 4. User Modelling

## 4.1 Introduction

Throughout the previous sections, we have often mentioned that knowledge about the user could aid a system in making various decisions required during the course of understanding and answering a question.

Rumelhart [54] and Shannon [62] were among the first researchers to recognize that some knowledge of the user was needed to generate appropriate answers. They showed that to answer *where questions*, knowledge about the user's location was important. Figure 4-1 shows three possible answers to a single where question.

```
Q:  Where is the Empire State Building?
A1: In the United States.
A2: In New York.
A3: On the corner of Fifth Avenue and 34th Street.

Each of these answers can be appropriate in the right context,
that is depending on where the questioner is (for example,
in Ivory Coast for A1, in Florida for A2, and in Manhattan for A3).
```

**Figure 4-1:** Knowledge about the user's location can influence an answer

Humans make use of their knowledge about other participants in a conversation in order to communicate effectively. It is clearly desirable for a computer system to have knowledge about the user in order to more closely approximate natural language question answering. In this section, we will examine various methods that have been developed to help achieve this goal.

A user model can contain a variety of facts about a user, including:

- The user's goals in asking a question. The goal can influence the way to address the question. As seen in the research done by Hobbs and Robinson, an appropriate answer is one that address the goal of the user. It is thus important to know what that goal is.
- The plan the user has to achieve the goal.
- The user's knowledge about the world. This will help in providing the appropriate information (that is, information that the user will understand).
- The type of the user, which can also influence the information given in the answer. In an information retrieval system for example, user types may include managers, secretaries, or engineers, each of whom should be addressed in a different manner.
- Information already given as responses to previous questions.

User modelling problems include the task of constructing a model, which can be done by either collecting facts from a user or inferring them from a dialog, that of organizing the model, and using it to improve the system's behavior in responding to questions and possibly to change its behavior if it becomes obvious that responses provided are not appropriate. In this chapter, we present some of the research done that addresses the various aspects of user modelling in question answering, showing how a man-machine interaction can be improved when a system is able to take a particular user into consideration.

## 4.2 Using knowledge about the user's plans and goals to generate helpful responses

Allen and Perrault [1] examined the problems of generating appropriate responses to questions by inferring the questioner's goal. In doing so, they also tried to be faithful to philosophical and linguistics accounts of speech acts [61]. They showed that, by keeping a model of the questioner's beliefs and by being able to infer plans and goals, a system can provide helpful and cooperative answers. They also developed a method that enables a system to derive the user's beliefs and goals. Using this method, a question answering system can build a user model that contains the user's goals and beliefs and use it to answer questions in a cooperative fashion. The types of cooperative answers a system would be able to generate using this user model included direct and indirect answers, as well as answers containing more information than was requested in the question.

Allen and Perrault considered speech acts in the context of a planning system [7]. Conversants have *models* of the world, which include beliefs about the world, beliefs about other conversants, and goals to achieve. Language is viewed as being goal-oriented: a speaker produces utterances in order to achieve an effect on the listener, typically modifying the listener's beliefs and goals. Upon observing a speech act, the listener is capable of inferring the speaker's goal and reconstructing a plan to achieve that goal. To be cooperative, the listener can now help the speaker achieve his goal by indicating what information is needed to achieve the goal. This can result in giving more information than explicitly asked for if the listener recognizes such information as being necessary to achieve the goal (example in Figure 4-2).

```
Patron: When does the train to Montreal leave?
Clerk : 3:15 at gate 7.

The clerk was able to infer the patron's goal to board  the
Montreal Train.   Upon reconstructing  the plan  needed  to
achieve that goal,  the clerk recognized  that knowing  the
departure gate was  also necessary  (besides the  departure
time, which was explicitly asked  for).  So the clerk  gave
these  two  pieces  of  information  in  the  answer,  thus
appearing cooperative.
```

**Figure 4-2:** Cooperative behavior: inferring the speaker's plan

To detect the user's goals and plans, a system needs domain knowledge that includes *plans* and *goals* users may have in the domain of discourse, a formulation of *actions*, which have preconditions, substeps and effects, and *beliefs* and *wants* (intentions). In their system, Allen and Perrault used a standard planning formalism to represent plans and goals [12], in which given an *initial state of the world W* and a goal *G*, a *plan* is a sequence of actions that transform *W* into *G*. Two simple plans in the train domain, as used by Allen and Perrault, are shown in Figure 4-3. Allen and Perrault's plans are similar to those described in [58] . Schank and Abelson, however, used plans as an aid in finding the relations between two acts in order to understand a story. Here, plans were used to derive the goal of a questioner and thus help achieve a goal. Schank and Abelson also developed the notion of goal, but the goals they used were rather general, while those used in Allen and Perrault are domain specific. Beliefs and intentions were represented as in [22, 7]. Because this knowledge is represented explicitly, the system is able to reason about what the user needs to know in order to achieve a plan. This fact is very important since a system appears to be cooperative when it is able to provide information that will help the user to achieve a goal. Allen and Perrault also specified two types of inference rules, that the system could use to derive the user's goal: *planning rules* to infer what plan a user may have to achieve a goal; and *inferencing rules*, which are used to infer a goal from an observed action (see Figure 4-4).

Upon observing an action (or hearing a speech act, in the case of a question answering system), the system tried to reconstruct the user's plan. This was done by deriving the user's goal from the observed action using *inferencing rules* and finding plans that would achieve the expected goals the system knew about (such as *BOARD* and *MEET* in the train domain) using the *planning rules*. Heuristics were used to rate the different plans and goals, so that eventually one plan and one corresponding goal would be chosen as the correct plan (goal). An example of a plan reconstruction from a sentence is given in part in Figure 4-5.

```
    Expected goals:

            BOARD(A, <train>, TORONTO)

            MEET(A, <train>,  TORONTO)

    Actions:
          - BOARD(agent, train, station):SOURCE(train, station)

            precondition: AT(agent, the x:DEPART.LOC(train, x)),
                                    the x:DEPART.TIME(train, x)

            effect: ONBOARD(agent, train)

          - MEET(agent, train, station):DEST(train, station)

            precondition: AT(agent, the x:ARRIVE.LOC(train, x)),
                                    the x:ARRIVE.TIME(train, x)

            effect: MET(agent, train)
```

**Figure 4-3:** The train domain

Helpful responses

Allen and Perrault claimed that helpful responses are needed when the listener detects a goal in the plan of the speaker that the speaker cannot achieve without assistance. Such goals typically correspond to some information the speaker lacks but needs to achieve his goal. In our previous example, knowing the departure gate is one such goal (if the listener believes that the speaker does not yet know it).

After reconstructing the user's goal and plan, the system would find the goals it believed the user needed help to achieve and produce plans to achieve them.[19] When they were achieved, the answer to a question was produced:

```
                Q: ''When does the train to Windsor leave?''
                A: ''The train  leaves at  4:00pm.   The train
    leaves at  gate  4.''
```

This led the system to include more information than explicitly requested.

[19]Note that a plan was executed for each goal, as opposed to executing one plan to accomplish several goals at once.

<u>Example of Planning rules</u>:

- If an agent A wants to achieve a goal E, and ACT is an action that has E as an effect, then A may want to execute ACT (i.e. achieve ACT).

- If an agent A wants to achieve P and does not know whether P is true or not, then A may want to achieve:

  ''agent knows whether P is true''.

<u>Example of Inference rules</u>:

- If S believes that A has a goal of executing action ACT, and ACT has an effect E, then S may believe that A has a goal of achieving E.

- If S believes A has a goal of knowing whether a proposition P is true, then S may believe that A has a goal of achieving P.

**Figure 4-4:** Planning and inferencing rules

Helpful information can also be offered in response to a Yes/No question. From the utterance:

''Does the Windsor train leave at 4?''

the listener can infer the goal:

The speaker wants to know when the train leaves

This goal, in turn, can be connected to the *BOARD* act, whose preconditions require knowing the departure time and gate. Here, if the direct answer to the question is "No", the listener, by inferring the goal of the speaker and recognizing obstacles in the plan, will provide useful information such as the actual departure time:

''No, the train leaves at 6pm.''

<u>Goals and indirect speech acts</u>
Plan inferencing allows for the understanding of indirect speech acts. By reconstructing the plan from the utterance, the indirect interpretation can be inferred. For example, from "Do you know when the Windsor train leaves", the goal "Agent wants to know the departure time" can be inferred. Allen and Perrault extend their theory and analysis in order to distinguish between the cases where a question is intended literally and when it is intended indirectly. To do so, they use Searle definition of *surface speech acts*, which correspond to the literal meaning of the utterance, and *illocutionary acts*, which correspond to the indirect meaning [61]. An *intentional* precondition is now added to detect whether an utterance should be interpreted directly or indirectly, and the rules are adjusted appropriately. Note that now, the process is complicated by the fact that the beliefs need to include the intentions of the conversant. Inferring these may be very hard.

<u>Understanding sentence fragments</u>
The plan inference process aids in the understanding of sentence fragments (such as "Train to Windsor?"), because such fragments are usually sufficient to infer the basic goal. Compare this approach to that taken in QUALM [36],

```
From the speech act:  ''When does the train to Windsor leave?''

a REQUEST action can be constructed[17]:

        REQUEST(A, S, INFORMREF(S, A, the (x:time):
                DEPART.TIME of <train1> is x))
               where <train1> = the (x:train:DEST(x, WINDSOR))

1. This action specifies  an action cluster consisting of  a
   REQUEST  of  INFORMREF[18].   It   is  added  to  the  plan
  alternative in each partial plan.

2. The partial plans are examined for similarities
   between alternatives and expectations:

   - BOARD plan:  A
                  the train to WINDSOR
                  the DEPART.TIME
   - MEET plan:   A

3. The BOARD plan is favored as it is more specified, and the
   train descriptions in the alternative and  the expectation
   are  merged. Now  both the source  and  the  destination
   are  known.

4. ...[More tasks are executed to complete the plan.]
```

**Figure 4-5:** Example of Plan reconstruction

where fragments were understood within a script or by using the discourse topic. Using plan inference rules is more applicable in some cases where a script is not available. On the other hand, the script knowledge could alleviate some of the processing involved in the required inferencing. In a system containing a large knowledge base, both approaches have the problem of identifying what the utterance refers to (which script to activate, or which action is involved). Using a script or plan inferencing allow a sentence fragment to be understood even before a topic of discourse has been established (or a previous query encountered).

Conclusions

Inferring the goal of a speaker and detecting obstacles in the speaker's plan provide a more general framework within which some linguistic phenomena can be explained. In particular, it provides a method of generating (and explaining) cooperative responses. Unfortunately, the inference process is quite lengthy, rendering this approach computationally expensive for large-scale applications. Furthermore, the system has to know ahead of time all the possible actions permissible in the domain. Adding domain dependent knowledge may be required in some cases. Further research on plans and goals and their use in cooperative discourse continues [65, 52, 4].

[17]The speech act definitions are based on [49]

[18]INFORMREF is to inform an agent of the referent of a variable.

## 4.3 Constructing a user model with the aid of stereotypes

Another type of user model is one that attempts to capture the characteristics of the user in terms of user types. Rich [53] studied how to construct such a model using stereotypes. Researchers have made use of stereotypes for both story understanding and story generation, as they seem to embody implicit but necessary information ( [59, 34]). Rich claimed that stereotypes can also be used in question-answering to build a user model and that humans themselves often construct such a model of a person based on stereotypes they know. As an example, when talking to a lawyer, people tend to assume the lawyer is highly educated and wealthy, as lawyers might be expected to be.

Rich showed how a model of the user can be built by refining and intersecting various stereotypes and how a system can then use this model to tailor its answer to a user. GRUNDY, a system simulating a librarian, made use of such a method to suggest books to its users. Although GRUNDY was not a question answering program, the method it employed to construct a user model is applicable to such systems.

GRUNDY used a generalization hierarchy of stereotypes, each containing a set of characteristics. Each characteristic was a triple including an attribute or *facet*, its value, and a rating. The stereotype for a *feminist* is shown in Figure 4-6. *Politics*, *Sex-open*, or *Tolerate-sex* are facets of this stereotype. Their values, ranging from -5 to 5 indicate the truth value of the facet, while the rating represents how confident the system is of the information.

| FACET | VALUE<br>(from -5 to 5) | RATING<br>(out of 1000) |
|---|---|---|
| Genl | ANY-PERSON | |
| Genres | | |
| woman | 3 | 700 |
| Politics | Liberal | 700 |
| Sex-open | 5 | 900 |
| Piety | -5 | 800 |
| Political-causes | | |
| Women | 5 | 1000 |
| Conflicts | | |
| Sex-roles | 4 | 900 |
| Upbringing | 3 | 800 |
| Tolerate-sex | 5 | 700 |
| Strength | | |
| Perseverance | 3 | 600 |
| Independence | 3 | 600 |
| *Triggers* | *Fem-woman-trig* | |

FEMINIST STEREOTYPE

**Figure 4-6:** Example of stereotype in GRUNDY

Associated with a stereotype were also *triggers* which signalled the appropriate use of a stereotype.

Stereotypes were activated through triggers when users were asked to describe themselves by typing a few words. Because of the generalization hierarchy, one stereotype could also activate another one: as an example, if the *Protestant* stereotype was activated, then the more general *Christian* stereotype would be activated as well. The user model was built up by combining the characteristics of the active stereotypes. When the same characteristic occurred in several stereotypes, the user model would contain the average of their values, weighted by their respective rating. It was hoped that, having several stereotypes active at one point would result in a reasonable picture of the user. The user model thus contained a set of characteristics, taken from the active stereotypes. A *justification* was associated with each characteristic. This justification indicated which stereotype the facet was borrowed from, in case the system needed to remember how the information was derived.

Once the model was built, the system used it to select a book to present to the user. The most salient characteristics of the user were selected, and one was chosen at random to serve as a basis for selection. As the objects in the knowledge base (books) also had attributes that corresponded to the facets of the users' stereotypes, a set of books was selected that matched the chosen characteristic. Each book of the set was then evaluated against the other salient characteristics of the user, and the best match was presented to the user.

In presenting the book to the user, GRUNDY used the information in the user model to decide on which aspects of the book to mention. When the book was refused, GRUNDY would try to understand why by asking the user which characteristic of the book was disliked. Based on the answer, GRUNDY would try to alter the user model by changing the inappropriate characteristic.

Validity of this approach and its use in question answering systems

User types are important in question answering systems for they provide a convenient way for systems to draw inferences on the kind of information to present to the user. In some sense, systems today already use the idea of a stereotype by the fact that their builders make some assumptions about the typical users of the system and design it for such users. Some systems recognize that there may be different types of users (typically only two, expert and naive), and associate some information in the system with each type [63]. However, the different types are not explicitly represented as in Rich's system.

Rich's approach has the advantage of offering more possible distinctions among users. Furthermore, because the system builds the individual user model dynamically by combining many static models, it allows for greater flexibility as to what characteristics are included in the user model and is thus better able to tailor a response to the user. In order to use this approach in a question answering program, one would have to be able identify the possible users of the system and characterize them.

Unfortunately, the stereotypes built into the system are totally dependent on the system builder's view of the possible users: the characteristics included in the stereotype can be colored by this view, and the numbers given as values to facets seem rather arbitrary. Users are not just an intersection and conglomerate of stereotypes, and it can be harmful to not recognize this fact. Furthermore, combining stereotypes by taking an average of their common characteristics oversimplifies the complex issue of relative stereotype and characteristic importance. Finally, even if the user model constructed in that fashion reflected the user's characteristics faithfully, it may not be a good idea to base the answer only on these characteristics.

Stereotypes are a good way to start a user model, but they should be complemented by other methods that would allow a model to be further specified and customized. For example, modifiers could be introduced. Furthermore, a system must be able to adapt itself and change the way it infers a model and uses it. For GRUNDY, this would mean being able to detect that the use of a stereotype is inappropriate in a certain situation (e.g., that a trigger is invalid), to recognize that some piece of information is not appropriate for a certain class of user (e.g., that the characteristic associated with an object is inaccurate), or that a stereotype contains invalid characteristics. This adaptation could be based on feedback from the user.

## 4.4 User modelling and focus in natural language interfaces to databases

Instead of employing user models to vary the amount of information given in answers, Davidson [11] used a user model to facilitate the interpretation of definite noun phrases. In PIQUE, Davidson showed how a user model containing *focus* information can be used to disambiguate sentences containing definite noun phrase in a database interface. When accessing a database, users typically ask several questions on a topic (on one part of the database), but do not follow a general conversational pattern, as they would if they were conversing with a person: a topic of conversation is not introduced, and there are no linguistic cues that indicate when a topic changes. As a result, the topic of conversation is not necessarily the crucial piece of information that allows the system to disambiguate questions. Instead, Davidson argued that a system needs to remember which aspect of the database (called the *focus*) was previously examined (see Figure 4-7). This *focus* is thus different from a discourse model. It refers to the part of the database a user was last interested in.

```
Q1: Who are the programmers?
A1: Jones, Smith, Baker
Q2: What's Jones' salary?
A2: There are 24 employees named 'Jones'.
    Which one do you mean?

In this example, it is clear that the user had in mind the
person that was just mentioned.  The system however did
not behave intelligently or cooperatively by not
recognizing that fact.  It would have been able to
understand the definite noun phrase had it remembered what
part of the database was just accessed.
```

**Figure 4-7:** Sample interaction with a database

In a database system, a user's database query is typically represented in a formal data manipulation language, usually a variant of relational calculus or algebra (example in Figure 4-8). When thus represented, a query can also be seen as an *intensional description* of some set of the database.

```
The query: Who are the programmers?

might be expressed as:

 { x.name : (x belongs to the set of employees) |
            x.occupation = programmer }

This expression can be viewed as an intensional description
of the set of programmers.
```

**Figure 4-8:** A database query

PIQUE used the segment of the database described by such a query to represent a user's focus. The user's focus was then used to interpret future inputs containing definite noun phrases, by providing a frame of reference. Based on the set of objects highlighted by the focus model, the system added more constraints to those queries, as shown in Figure 4-9.

```
From the query Q1: ''Who are the programmers?''
which is represented as:

      { x.name : (x belongs to the set of employees) |
                   x.occupation = "programmer" }

PIQUE sets the focus information to be:
    the set of employees which are programmers

A future query might be:
        Q2: ''What is Jones' salary?''

Q2 is represented as:

      { x.sal : (x belongs to the set of employees) |
                x.name = "Jones" }

Using the focus model, the system adds constraints to the
representation of Q2 to get:

      { x.sal : (x belongs to the set of employees) |
                x.name = "Jones"
                AND x.occupation = "programmer"}

Using this representation, Q2 can be correctly understood as:

'' What is Jones' salary, where Jones is one of the programmers''
```

**Figure 4-9:** Using focus information to add constraints

To make sure the user was aware of the system's interpretation of a query, PIQUE informed the user when the user focus was used and gave the assumed referent of an expression. Like paraphrase, this feedback avoided confusion, in case the query was misinterpreted.

This simple focus model is significant because of its relation to the database concept of *views*, which represent the way the users see the relations in the database. Furthermore, the focus model was obtained without any extra processing.

## 4.5 Conclusions

Using information about the user, computer systems can treat users as individuals and address their particular needs. Adding information about the user thus potentially renders a question-answering system more useful and leads to major progress towards the goal of graceful interaction. Researchers have started to study the problem of user modelling and its use on question-answering, as we have seen in this section: by inferring the goals of a user, a system can generate helpful (possibly indirect) responses; the type of a user can be used to decide on the kind of information to provide; finally, by keeping the "user's view" of the context of a question, a system can understand definite noun phrases entered by the user.

However, the study of user modelling and its importance is a fairly recent area of study, and work continues on the development of user models that can guide a system in generating answers containing ellipsis and anaphora [27], in generating the appropriate response [47, 48] or explanation [46] to a user and in detecting and correcting misconceptions [60]. Many issues in user modelling are still under study, including the representation of a user model, its use in a system and how to update it if necessary.

## 5. Conclusions

We have surveyed a variety of methods used to provide more graceful interaction between man and machine and more closely approximate human question answering, from human engineering features to the development of language theories. The common goal of the programs surveyed was to provide a natural language interface that would allow users to express themselves as freely as possible and that would answer questions as a human would. Each of the theories studied was applied in a particular domain and each represented a subset of what is needed to fully achieve human question answering abilities and thus true graceful interaction. The following table summarizes the progress made towards graceful interaction and which aspects of graceful interaction were addressed by the systems (or theories) presented:[20]

| | Flexible Input | Data Independence | Cooperation | Avoidance of Confusion |
|---|---|---|---|---|
| BASEBALL | | | | |
| SYNTHEX[21] | | | | |
| LUNAR | XXX | | | |
| LIFER | XXX | | | XXX [with feedback] |
| PLANES | XXX | | | XXX [with feedback] |
| QUALM | | | XXX | |
| TEXT | | XXX | XXX | |
| CO-OP | | XXX | XXX | XXX |
| Mays' system | | XXX | XXX | XXX |
| Hirshberg | | | XXX | |
| Allen and Perrault | | | XXX | |
| GRUNDY | | | XXX | |
| PIQUE | | | XXX | |

Further research is needed to study how these theories could interact with each other, complement and help each other to make up more comprehensive systems. More basic work is also needed however. None of the problems presented have been fully solved, and some problems of question answering have not been addressed at all. To be truly cooperative, a system should be able to ask the user questions when additional information is needed. In order to do so, one must study when it is appropriate to pose questions, and how the system should do it to be most effective [39]. For a user model to be complete, it must include some information about the user's knowledge about the domain of discourse. Tailoring a response based on this knowledge is an important part of user modelling [48]. Learning is also an important issue: a system should be able to learn from past experiences: that is, it should be able to improve itself based on its previous performance. Finally, the problem of generation (actually producing

[20] Each of these aspects can involve several issues and have been solved using different approaches as described throughout this paper.

[21] The problem of graceful interaction was not really addressed yet; these programs were among the first programs to allow questions in English.

output in good English) has been studied in some systems ( [42, 38, 45]), but further work is needed. Research is being conducted in all these areas and researchers are applying and extending the methods we presented to other types of systems, such as expert systems, help systems and CAI systems.

## References

1. Allen, J. F. and Perrault, C. R. "Analyzing Intention in Utterances". *Artificial Intelligence 15*, 1 (1980), pages 143 - 178.

2. Belnap, N. and Steel, T.. *The Logic of Questions and Answers*. Yale University Press., New Haven, 1976.

3. Brown, J. S. and Burton, R. R. "Diagnostics Models for Procedural Bugs in Basic Mathematical Skills". *Cognitive Science 2 (2)* (1978), pages 155 - 192.

4. Carberry, S. Tracking User Goals in an Information-Seeking Environment. Proceedings of AAAI-83, American Association of Artificial Intelligence, 1983.

5. Codd, E. F. Seven Steps to Rendez-vous with the Casual User. In J. W. Klimbie and K. Koffeman, Ed., *Data Base Management*, North-Holland, Amsterdam, 1977.

6. Codd E. F., Arnold R. S., Cadiou J-M., Chang C. L. and Roussopoulos N. Rendezvous Version 1: An Experimental English-Language Query Formulation System for Casual Users of Relational Data Bases. Research Report RJ2144(29407), 1/26/78, IBM Research Laboratory, 1978. Computer Science, Research Division, San Jose, Ca.

7. Cohen, P. On Knowing What to Say: Planning Speech Acts. 118, University of Toronto, January, 1978. Department of Computer Science.

8. Cohen, P. R. and Perrault, C. R. "Elements of a Plan-Based Theory of Speech Acts". *Cognitive Science 3* (1979), pages 177 - 212.

9. Cullingford R. E. Script Application: Computer Understanding of Newspaper Stories. Research Report 116, Yale University, Department of Computer Science, 1978.

10. Damerau, F. Advantages of a Transformational Grammar for Question Answering. Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 1977.

11. Davidson, J. Natural Language Access to Data Bases: User Modelling and Focus. Proceedings of the Fourth Biennial Conference of the CSCSI, Canadian Society for Computational Studies of Intelligence', Saskatoon, Saskatchewan, 1982, pp. 204 - 211.

12. Fikes, R. E. and Nilsson, H. J. "STRIP: A New Approach to the Application of Theorem Proving to Problem Solving". *Artificial Intelligence 2* (1971), pages 189 - 205.

13. Gadzar, G.. *Pragmatics: Implicature, Presupposition and Logical Form.* Academic Press, New York, 1979.

14. Green, B.F., Wolf, A.K., Chomsky, C and Laughery, K. Baseball: An Automatic Question-Answerer. In Feigenbaum, E.A. and Feldman, J., Ed., *Computers and Thought*, McGraw-Hill, New York, 1963.

15. Grice, H. P. "Utterer's Meaning and Intentions". *Philosophical Review 68*, 2 (1969), 147 - 177.

16. Grice, H. P. Logic and Conversation. In *Syntax and Semantics*, P. Cole and J. L. Morgan, Ed., Academic Press, New York, 1975.

17. Grimes, J. E.. *The Thread of Discourse*. Mouton, The Hague, 1975.

18. Grosz, B. J. The Representation and Use of Focus in a System for Understanding Dialogs. Proceedings of the IJCAI, Pittsburgh, Pennsylvania, 1977.

19. Grosz, B. J. Utterance and Objective: Issues in Natural Language Communication. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, IJCAI, Tokyo, Japan, 1979.

20. Hayes, P. and Reddy, R. Graceful Interaction in Man-Machine Communication. Proceedings of the IJCAI, 1979.

21. Hendrix, G. G. Human Engineering for Applied Natural Language Processing. Proceedings of IJCAI-77, International Joint Conferences on Artificial Intelligence, Cambridge, MA, 1977.

22. Hintikka, J.. *Knowledge and Belief.* Cornell University Press, Ithaca, New York, 1963.

23. Hirschberg, J. Towards a Redefinition of Yes/No Questions. Proceedings of the 22nd Annual Meeting of the ACL, Association for Computational Linguistics, Stanford University, California, 1984.

24. Hobbs, J. R. and Robinson, J. J. Why Ask? Technical Note 169, SRI International, October, 1978.

25. Horn, L. R. On the Semantic Properties of te Logical Operators in English. Mimeo, Indiana University Linguistics Club.

26. Horn, L. R. Greek Grice. Proceedings of the Ninth Regional Meeting of the Chicago Linguistics Society, Chicago Linguistics Society, Chicago, Illinois, 1973.

27. Jameson, A. and Wahlster, W. User Modelling in Anaphora Generation: Ellipsis and Definite Description. Proceedings of ECAI-82, ECAI, Orsay, France, 1982, pp. 222 - 227.

28. Joshi, A. K. Mutual Beliefs in Question-Answer Systems. In N. Smith, Ed., *Mutual Beliefs*, Academic Press, New York, 1982.

29. Joshi, A. , Webber, B and Weischedel, R. Living Up to Expectations: Computing Expert Responses. Proceedings of AAAI-84, American Association of Artificial Intelligence, 1984.

30. Kaplan J. S. Indirect Responses to Loaded Questions. Proceedings of the Second Workshop in Theoritical Issues in Natural Language Processing, Theoritical Issues in Natural Language Processing, Urbana-Champaign, Illinois, 1978.

31. Kaplan, S. J. *Cooperative Responses from a Portable Natural Language Database Query System.* Ph.D. Th., University of Pennsylvania, Philadelphia, Pa, 1979.

32. Kaplan, S. J. "Cooperative Responses from a Portable Natural Language Query System.". *Artificial Intelligence* 2, 19 (1982).

33. Keenan, E. L. Two Kinds of Presuppositions in Natural Language. In *Studies in Linguistic Semantics*, C. J. Fillmore and D. T. Langendoen, Ed., Holt, Rinehart, and Winston, New York, 1971.

34. Lebowitz, M. "Creating Characters in a Story-Telling Universe". *Poetics 13* (1984), 171 - 194.

35. Lehnert, W. A Conceptual Theory of Question Answering. Proceedings of the IJCAI-77, Cambridge, Massachusetts, 1977.

36. Lehnert, W. G.. *The Process of Question Answering.* Lawrence Erlbaum Associates, Hillsdale, N. J., 1978.

37. Levinson, S. C.. *Pragmatics.* Cambridge University Press, Cambridge, England, 1983.

38. Mann, W. C., Bates M., Grosz B. J., McDonald, D. D., McKeown, K. R., and Swartout, W. R. Text Generation: The State of the Art and the Literature. ISI/RR-81-101, Information Science Institute, 1981. Also a Technical Report at the University of Pennsylvania, number MS-CIS-81-9.

39. Matthews, K. Taking the Initiative in Problem-Solving Discourse. CUCS-114-84, Columbia University, 1984.

40. Mays, E. Correcting Misconceptions about Data Base Structure. Proceedings 3-CSCSI, Canadian Society of Computational Studies of Intelligence, Victoria, B. C., May, 1980.

41. McCoy, K. F. Correcting Misconceptions: What to say When the User is Mistaken. Proceedings of the CHI'83; Conference on Human Factors in Computing Systems, Human Factors, Boston, Ma, 1983.

42. McGuire, R. Political Primaries and Words of Pain. Yale University Department of Computer Science, 1980.

43. McKeown, K. R. Paraphrasing Using Given and New Information in a Question-Answer System. Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, La Jolla, Ca., August, 1979, pp. 67-72.

44. McKeown, K. *Generating Natural Language Text in Response to Questions About Database Structure.* Ph.D. Th., University of Pennsylvania, May 1982. Also a Technical report, No MS-CIS-82-05, University of Pennsylvania, 1982.

45. McKeown, K. R. The TEXT System for Natural Language Generation: An Overview. Proceedings of the 20th Annual Meeting of the ACL, Toronto, June, 1982.

46. McKeown, K. R., Wish, M. and Matthews, K. Tailoring Explanations for the User. Proceedings of the IJCAI, 1985.

47. Paris, C. L. Determining the Level of Expertise. Proceedings of the First Annual Workshop on Theoretical Issues in Conceptual Information Processing, Atlanta, Georgia, 1984.

48. Paris, C. L. Description Strategies for Naive and Expert Users. Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, Chicago, 1985.

49. Perrault R. C. and Allen J. F. "A Plan-Based Analysis of Indirect Speech Acts". *American Journal of Computational Linguistics 6*, 3-4 (1980).

50. Petrick, S. R. Transformational Analysis. In *Natural Language Processing*, R. Rustin, Ed., Algorithmics Press, New York, 1973.

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

52. Pollack, M., Hirschberg, J. & Webber, B. User Participation in the Reasoning Processes of Expert Systems. Proceedings of the AAAI, American Association of Artificial Intelligence, Pittsburgh, Pa, 1982.

53. Rich, E. A. "User Modeling Via Stereotypes". *Cognitive Science 3* (1979), 329 - 354.

54. Rumelhart, D. The Room Theory. The University of California at San Diego.

55. Sacerdoti, E. Language Access to Distributed Data with Error Recovery. Proceedings of the IJCAI-77, Cambridge, Massachusetts, 1977.

56. Schank, R. C. "Conceptual Dependency: A Theory of Natural Language Understanding". *Cognitive Psychology 3*, 4 (1972), pages 532 - 631.

57. Schank, R. C.. *Conceptual Information Processing*. North Holland, Amsterdam, 1975.

58. Schank, R. C. and Abelson, R. P.. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

59. Schank, R. C. and Lebowitz, M. Does a Hippie Own a Hairdrier? 144, Yale University Department of Computer Science, 1979.

60. Schuster, E. Custom-Made Responses: Maintaining and Updating the User Model. ms-83-13, Unversity of Pennsylvania, Department of Computer and Information Science, September, 1983.

61. Searle, J. R. Indirect Speech Acts. In *Syntax and Semantics*, P. Cole and J. L. Morgan, Ed., Academic Press, New York, 1975.

62. Shannon, B. Where Questions. Proceedings of the ACL-79, Association of Computational Linguistics, 1979.

63. Shortliffe, E. H., Davis, R., Buchanan, B. G., Axline, S. G., Green, C. C., and Cohen, S. N. "Computer-Based Consultations in Clinical Therapeutics - Explanation and Rule Acquisition Capabilities of the MYCIN System". *Computers and Biomedical Research 8* (1975), 303 - 320.

64. Sidner, C. L. *Towards a Computation Theory of Definite Anaphora Comprehension in English Discourse* . Ph.D. Th., MIT, cambridge, MA, 1979.

65. Sidner, C. and Israel D. Recognizing Intended Meaning and Speakers' Plans. Proceedings of the IJCAI, August, 1981.

66. Simmons, R.F., Burger, J.F. and Long, , B.E.. An Approach toward Answering Questions from Text. Proceedings of the Fall Joint Computer Conference, AFIPS Press, Montvale, N.J., 1966.

67. Sleeman, D. Inferring (Mal) Rules From Pupil's Protocols. Proceedings of ECAI-82, European Conference (?) of Artificial Intelligence, Orsay, France, 1982.

68. Stevens, A. , Collins, A. Goldin, S. E. "Misconceptions in Student's Understanding.". *International Journal of Man-machine Studies 11* (1979), pages 145 - 156.

69. Tennant H. The Evaluation of Natural Language Question Answerers. University of Illinois at Urbana/Champaign, 1978. Ph.D. Proposal, Department of Computer Science, Advanced Automation Group, Coordinated Science Laboratory.

70. Tennant, H.. *Natural Language Processing.* Petrocelli Books, Inc., 1981.

71. Tennant, H. R., Ross, K. M., and Thompson C. W. Usable Natural Language Interface Through Menu-Based Natural Language Understanding. CHI'83 Proceedings, 1983.

72. Ullman, J. D.. *Principles of Database Systems.* Computer Science Press, Rockville, Maryland, 1980.

73. Waltz, D. L. and Goodman, B. A. Writing a Natural Language Data Base System. Proceedings of IJCAI-77, International Joint Conferences on Artificial Intelligence, 1977.

74. Wilensky, R.. *Planning and Understanding.* Addison-Wesley, Reading, MA, 1983.

75. Woods, W. A. Procedural Semantics for Question-Answering Machine. Proceedings of the Fall Joint Computer Conference, Montvale, N.J., 1968.

76. Woods, W. An Experimental Parsing System for Transition Network Grammars. In *Natural Language Processing*, Rustin, R., Ed., Algorithmics Press, New York, 1973.

77. Woods, W. A. and Kaplan, R. M. The Lunar Sciences Natural Language Information System: Final Report. BBN Report 2265, Bolt Beranek and Newman, Inc., Cambridge, MA, 1972.