Comp. & Maths. with Appls. Vol. 9, No. 1, pp. 233–244, 1983
Printed in Great Britain.

0097-4943/83/010233-12$03.00/0
Pergamon Press Ltd.

EXTENDED NATURAL LANGUAGE DATA BASE INTERACTIONS

BONNIE WEBBER, ARAVIND JOSHI, ERIC MAYS and KATHLEEN MCKEOWN
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, U.S.A.

Abstract—An oft-heard argument for Natural Language interfaces is their promise of reducing the effort an infrequent user would have to exert in using a computer system. This viewpoint has led to a research concentration on removing "artificial" constraints on a user's freedom of expression, allowing it to move closer to everyday speech. This paper discusses two complementary directions for extending Natural Language interfaces to data bases, which can make them more useful systems. These extensions make Natural Language interfaces less simply "windows" through which data can be called into view, and more "articulate experts" on the data base system and what it represents. The two directions involve (1) broadening the range of *query types* that can be handled and (2) extending the range of *responses* that can be provided.

1. INTRODUCTION

If one considers the history of Natural Language interaction with computers, one is called to mind of the strongest reason given for wanting to use Natural Language: it promised to reduce the effort an infrequent user would have to exert in using a computer system. S/he would not have to learn and retain a host of different languages and protocols. S/he would be able to express his/her desires for information directly.

As Natural Language (NL) interfaces have been viewed in terms of this promise, their primary direction of development has been towards removing "artificial" constraints on a user's freedom of expression, allowing it to move closer to everyday speech. Developments along this line include:

- Enabling NL systems to interpret a user's query in light of his/her previous queries or of his/her perceived intentions. This permits the user to produce elliptic utterances and anaphoric expressions that will be understood by the system [1–3].
- Enabling NL systems to recognize the user's intended question, even though it might contain mispellings, simple grammatical errors, etc. [2, 4].
- Enabling users to set up "on line" whatever terminology they want to use in interacting with the system, for example by typing something like "By ⟨x⟩ I mean ⟨y⟩", where ⟨y⟩ is a form already interpretable by the system [2].

But just expanding the ways a query can be phrased and still understood does not in itself guarantee fruitful Natural Language interactions. What we shall discuss in this paper are two complementary directions for extending Natural Language interfaces to data bases, which can make them more useful systems. These extensions make Natural Language interfaces less simply "windows" through which data can be called into view, and more "articulate experts" on the data base system and what it represents. The two directions involve (1) broadening the range of *query types* that can be handled and (2) extending the range of *responses* that can be provided. More specifically, Section 2 describes a prototype system TEXT that is able to answer queries about the *structure* of a data base, and Section 3 discusses various ways a system may "take the initiative" and provide more than just a direct response to the user's query.

2. RESPONDING TO QUERIES ABOUT DATA BASE STRUCTURE

In asking a factual data base query, a user is seeking to establish either the existence or the identity of some restricted class of objects in the database. However, to ask such a question *and* have it make sense to the system, the user must already know what information is stored in the database and how it is structured. (Even if s/he does know what type of information is

available, its structure in the database may not correspond to his or her conception of it and again, his or her questions may not be understood by the system.) The problem lies in acquiring such knowledge. Up to now, there has been no way for a user to request it from the system. The only recourse has been to the data base administrator or to documentation, which is usually either missing, out-of-date or incomprehensible.

In this regard, a number of interesting experiments have been conducted to find out just what kinds of abilities a data base system interface should ideally have [5, 6]. In these experiments, users were given a problem to solve and a data base system with which they could freely interact in Natural Language. (This was usually facilitated through another person hidden behind the scenes.) A user's goal was to acquire enough information from the data base to diagnose and solve the problem.

What is of interest is that factual questions about the contents of the data base were not the only kind asked. Before attempting the problem solving task, users commonly worked at familiarizing themselves with the data base system and what it could do for them. In addition, in the midst of problem solving, users frequently attempted to confirm their understanding of what they had learned. In both cases, interactions with the system included (as illustrated below) various types of questions about the *structure* of the data base as well as its contents: (1) requests for definitions, (2) questions about the kind of information available in the database, and (3) questions about the difference between entities existing in the database.†

(1) What do you know about unit cost?
(2) What kind of data do you have?
(3) What is the difference between material cost and production cost?

Providing "canned" (i.e. hand-coded) answers to such questions is not an ideal solution for many reasons, including the fact that for "difference" questions, one would have to can a response for each possible pair of concepts. More importantly, such canned responses can easily become out-of-date if the data base changes without their prompt re-adjustment as well.

In this section, we shall report briefly on a system, *TEXT* [7], that was developed and implemented to respond dynamically to the three classes of questions mentioned above. Responding to such questions requires more than a simple search of the data base, as it is rarely clear *a priori* just what information is sufficient to answer them. Moreover, such questions can rarely be answered in a single sentence. Thus a system must be able to determine not only what information is appropriate to include in the answer but also how to organize it effectively into a multi-sentential text.

TEXT uses general principles of discourse structure, discourse coherence and relevance in generating responses to queries. Its main features include (1) an ability to identify and select that information that is potentially relevant to the answer, (2) an ability to pair rhetorical techniques (such as analogy) with discourse purposes (such as providing definitions) and (3) the use of a focusing mechanism to both choose among the potentially relevant information and produce coherent texts.

The notion of *rhetorical technique* comes from linguistics [8], as a way of encoding aspects of discourse structure. Standard rhetorical techniques include *attributive* (i.e. association of properties with an entity or event), *analogy* (i.e. comparison with a familiar concept), and *constituency* (i.e. presentation of an entity's sub-parts or sub-classes). In TEXT these techniques are used to guide the selection of propositions from a *relevant knowledge pool*—the subset of its knowledge base which TEXT has identified as containing all information which can be included in the answer. TEXT's knowledge base is made up of taxonomic, functional and attributive information about the concepts in the data base. Currently, it is in part hand-coded and in part compiled automatically from the facts in the data base [9]. The *focusing mechanism* draws on work of Grosz [10] and Sidner [11] and is used to maintain discourse coherency, both

†Users asked other types of questions as well, including "why" questions, "can" questions both about the domain (e.g. "What courses can a freshman register for?") and about the system (e.g. "What functions can you compute?"), and confirmation questions (e.g. "Do you mean item cost and unit cost are the same?"). The research reported here does not cover system support for such queries directly.

by (1) constraining the selection of information to be talked about next to that which ties in appropriately with the previous discourse and by (2) influencing what sentence-level syntactic structures are chosen, preferring those which support the desired information flow. A propos the first function, what can be said next is limited via the application of an *ordered* set of legal focus moves developed especially for automatic text generation and embodied in TEXT.

The current TEXT system is implemented on a portion of a data base supplied by the Office of Naval Research (ONR), containing information about vehicles and weapons. Two examples of TEXT system responses to questions about this domain are shown below. The first is a response to a request for a definition of a ship, while the second constitutes TEXT's answer to a request to distinguish between an ocean escort and a cruiser. (TEXT assumes a parser capable of interpreting such Natural Language requests as "What is a ship?", "What is the difference between . . . ?" into the input formalism shown below. Such a parser is not part of TEXT itself.)

1) (definition SHIP)
;
; What is a ship?
;
A ship is a water-going vehicle that travels on the surface. Its surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. Other DB attributes of the ship include MAXIMUM_SPEED, PROPULSION, FUEL (FUEL_CAPACITY and FUEL_TYPE), DIMENSIONS, SPEED_DEPENDENT_RANGE and OFFICIAL_NAME. The DOWNES, for example, has MAXIMUM_SPEED of 29, PROPULSION of STMTURGRD, FUEL of 810 (FUEL_CAPACITY) and BNKR (FUEL_TYPE), DIMENSIONS of 25 (DRAFT), 46 (BEAM), and 438 (LENGTH) and SPEED_DEPENDENT_RANGE of 4200 (ECONOMIC_RANGE) and 2200 (ENDURANCE_RANGE).

2) (difference OCEAN-ESCORT CRUISER)
;
; What is the difference between an ocean-escort and a cruiser?
;
The cruiser and the ocean-escort are surface ships. The ship's surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. The ship has DB attributes MAXIMUM_SPEED, PROPULSION, FUEL (FUEL_CAPACITY, AND FUEL-TYPE), DIMENSIONS, SPEED_DEPENDENT_RANGE and OFFICIAL-NAME. Ocean escorts have a DISPLACEMENT between 3400 and 4100. All ocean escorts in the ONR data base have REMARKS of 0, FUEL_TYPE OF BNKR, FLAG of BLBL, MAST_HEIGHT of 85 and PROPULSION of STMTURGRD. Ocean escorts carry between 2 and 22 torpedoes, 16 missiles and between 1 and 2 guns. A ship is classified as an ocean escort if the characters 1 through 2 of its HULL_NO are DE. Cruisers have a PROPULSION of STMTURGRD and a LENGTH between 510 and 673. All cruisers in the ONR database have REMARKS of 0 and FUEL_TYPE of BNKR. Cruisers carry between 8 and 42 torpedoes, between 4 and 98 missiles and between 1 and 4 guns. A ship is classified as a cruiser if the characters 1 through 2 of its HULLNO are CL or the characters 1 through 2 of its HULLNO are CG. The ocean escort, therefore, has a smaller LENGTH and a smaller DISPLACEMENT than the cruiser.

The structure of these responses reflect communicative strategies that are widely used in conveying similar information[7]. For example, people frequently *identify* an object through some combination of: (1) identifying it as a member of some generic class, (2) describing its function, attributes, and constituency (either physical or class), (3) making an analogy to a more familiar object, and (4) giving examples of the object. People rarely use these techniques in random order, commonly identifying an item as a member of some generic class before providing examples, etc.

In TEXT rhetorical techniques such as these are encoded as schemas which guide the generation process. The schema used for the identification of an object is shown below. ("/" separates alternatives; "*" indicates 0 or more instances; "+" indicates 1 or more instances; and curly brackets indicate optionality.)

Identification (class & attribute/function)
{Analogy/Constituency/Attributive/Renaming}*
Particular-illustration/Evidence +
{Amplification/Analogy/Attributive}
{Particular illustration/Evidence}

In the first example above, a ship is *identified* as belonging to the *class* of water-going vehicles, *evidence* for its surface-going capabilities are provided, and additional data base *attributes* of the ship are noted. The example concludes by noting the Downes as a particular illustration of a ship.

In concluding this section, we want to emphasize that by giving the user the ability to get direct answers automatically to such questions as these, we have taken a good-size step towards more mutually-intelligible user-system interactions. The development of TEXT is continuing in several different directions, including (1) extending the range of structural questions it can handle (e.g. allowing it to answer both types of "can" questions noted above); (2) applying it to dynamic as well as static data base systems; (3) expanding the range of Natural Language constructions it is able to use[12]; and (4) giving it an ability to present its response in multi-paragraph texts instead of a single paragraph. All of these features will make a system such as TEXT a more useful addition to the facilities available in Natural Language interfaces.

3. TAKING THE INITIATIVE IN RESPONDING TO QUERIES

As noted earlier, in answering a factual query one can often provide more than just the answer explicitly requested. We have been looking at three ways in which a system could so "take the initiative" in constructing a response: (1) it can point out incorrect assumptions reflected in the user's query[13, 14]; (2) it can offer to monitor for information that it may later learn of[15–17]; and (3) it can provide grounds for its response[18]. The following responses illustrate "presupposition correction", "monitor offers" and "justification", respectively. ("U" refers to the user and "S" to the system.)

U: Did John take Physics 161 last term?
S1: No, John can't take any courses: he's not a student.
S2: No, shall I let you know if and when he does?
S3: No, he advance-placed Physics 161.

This section will discuss some of the problems involved in supporting such "initiative-taking" behavior and some techniques for dealing with them.

3(a) *Presupposition correction*

It has been widely noted that for conversants to interact intelligibly, each must have a good sense of the other's beliefs and viewpoint. Achieving this mutual understanding can come about in two different ways: (1) *after* the fact, after some remark can be seen to reflect an incorrect understanding of the other's beliefs or viewpoint, or (2) *before* the fact, in that the conversants already know or have taken steps to learn about each other's beliefs and viewpoint. In interacting with a computer system, a user can learn about these things beforehand by asking the kinds of structural questions that TEXT has been developed to handle. The "after the fact" situation is discussed here.

In certain cases, it is not difficult to identify those things a user would have to believe before sensibly asking a particular question. For example, in asking the question "Which French majors failed CIS531 last term?" a user would have to believe: (1) CIS531 was given last term, (2) there are French majors, and (3) there are French majors who took CIS531 last term. In asking the question "Which graduate students have taught CIS531?" a user would have to believe that graduate students can teach CIS531. Any direct answer to such a question would implicitly confirm these beliefs, including the answer "None". Thus if the system can recognize that any of these beliefs are wrong, it shouldn't just give a direct answer, but rather correct the misconceptions as well.

Computational work on this problem includes Kaplan's work on correcting beliefs which "fail extensionally"[13] by supposing there to be entities which satisfy a particular description when there are in fact none and Mays' work on correcting beliefs which "fail intensionally" by supposing that a relationship can hold between two objects or classes, when it in fact cannot[14]. The kind of interactions this work supports is illustrated below:

U: Which French majors failed CIS531 last term?
S1: I do not know of any French majors.
S2: I do not know of CIS531 being given last term.
U: Which graduate students have taught CIS531?
S: CIS531 is a graduate course. Only faculty can teach graduate courses.

The techniques supporting such interactions have been well-documented elsewhere. They rely on (1) the system's ability to keep tabs on the non-emptiness of set extensions while it is doing a data base retrieval and (2) its having an enriched data model which indicates whether subsets are mutually exclusive and/or exhaustive. Rather than describe these techniques in detail here, we shall mention one direction in which this work on correcting misconceptions can go.

In Kaplan and Mays' work, an object's features (e.g. properties, class membership, etc.) are assumed to be fixed. From this point of view, presupposition failures are similar to "type violations" in a programming language. But most data bases are subject to update, so an object's features *can* change. However, not all systems can access their knowledge of possible changes: they may only apply it mechanically as update constraints. Alternatively, a system can be aware of how it has changed and how it can change—of what was true in previous states of the data base, of what is true now, and what can and cannot become true in the future. Not only is the data base subject to updates, the data base system remembers previous items and values and can reason about future ones. We have termed this a "dynamic data base" system[15–17].

To return to users' misconceptions, one reason that a user may be mistaken is that s/he is unaware of some change. In this case, a dynamic data base system may be able to provide more information in correcting misconceptions than a static one can. For example, a static system can only determine that John is not now a student and can therefore only provide the following answer to the question "What courses is John registered for?"

S: John cannot register for any courses: he's not a student.

However, a dynamic data base system may be able to determine that while John is not currently a student, he once was. If so, it can make the more informative response:

S: John flunked out last year and is no longer a student. Thus he cannot register for any courses.

Joshi[19] uses this type of situation (i.e. where the discourse participants are not equally up-to-date on relevant aspects of their world) to illustrate a type of interaction he calls "squaring away" beliefs. He shows how one participant can sense the need for such an interaction based on knowledge of mutual beliefs and either the possibility for or an awareness of a misconception on the other participant's part.

To conclude, the search for such historical information may be motivated by trying to establish the reason for the user's mistaken impression, or it may be motivated by a desire to "take the initiative" in a way that we shall describe in the next section, i.e. offering to *monitor* for information.

3(b) *Monitor offers*

Dynamic data base systems, mentioned above in the context of presupposition correction, enable another significant type of initiative-taking behavior. That is, in addition to answering a user's question on the basis of the information the system *currently* contains, a dynamic system can offer to monitor for *additional* information which it will provide to the user if and when it

learns of it. This enables the system to perform two useful functions: (1) to provide the user with additional information s/he has not explicitly requested, as in the first example below, or (2) to defer its response until it can give a better one, as in the second example.

U: Did John take CSE110 last term?
S: No. Do you want me to let you know if he does take it?
U: Did John pass CSE110?
S: No, the semester hasn't ended, so he hasn't received a grade yet. Shall I let you know then if he has passed CSE110?

To perform these useful services, a system must be able to recognize what events are really possible or what additional information it might really acquire. Otherwise, it might not distinguish between the following two situations, in which the same response is clearly not appropriate:

U: Is the JFK within 30 miles of L.A.?
S: No, but shall I let you know when it is?
U: Is Santa Cruz within 30 miles of L.A.?
S: No, but shall I let you know when it is?

In addition, a system must be able to identify when it is appropriate to perform such services: at a later time, the information may not be of *interest* to the user. In the following sections, we discuss both these issues.

(1) *Computing possible futures.* There are many things that will never be the case in the future because of what is true now or what was true in the past. If John is 30 now, he may turn 35 one day, but he can never become 20. If John received a grade in a course once, he can never register for that course again. In such cases, the system should not offer to inform the user if and when John turns 20 or registers for that course. In addition there are many things that are not true now and cannot become true unless some other event takes place. If John is not registered for a course, he cannot receive a grade for it until some time after he does register. In such cases, the system should not offer to inform the user if and when John receives a grade unless it first offers to inform the user if and when he registers for the course.

To compute whether an event can possibly occur or some state ever hold, a reasoning system is needed which can handle the notion of possible change. We are using a branching time temporal logic[20], which has the advantage over other temporal logics of being able to reason about contingent events, not just whether or not one event happened before/after another. In this logic, there are six composite temporal operators, made up of one operator on branches (E, for "some" branch, and A, for "all" branches) and one operator on times/states within a branch (X, for the next state after now on a branch; F, for now or some state after now on the branch; and G, for all states on the branch.) The composite operators have the following intuitive meanings:

$(EX)P$—holds iff P is true at some immediate future.
$(AX)P$—holds iff P is true at every immediate future.
$(EF)P$—holds iff P is true at some time of some future.
$(AF)P$—holds iff P is true at some time of every future.
$(EG)P$—holds iff P is true at every time of some future.
$(AG)P$—holds iff P is true at every time of every future.

To illustrate the use of these operators in specifying how the data base contents may or may not change, consider representing that portion of a university data base dealing with students passing courses and students registering for courses. Let the propositional variables P and R mean "student has passed course" and "student is registered for course", respectively .Then, among the axioms specifying the relationship of the current state of the data base to possible future states might be the following:

(1) $(AG)[P \rightarrow (AX)P]$—once a student has passed a course it remains the case that s/he has passed.

(2) $(AG)[(-P \ \& \ -R) \rightarrow (EX)R]$—if a student neither has passed a course nor is registered for it, then it is next possible that s/he is registered.

(3) $(AG)[R \rightarrow (EX)P]$—if a student is registered for a course then it is next possible that s/he has passed it.

(4) $(AG)[R \rightarrow (EX)[-P \ \& \ -R]]$—if a student is registered for a course then it is next possible that s/he neither has passed it nor is registered for it (i.e. s/he may drop it).

(5) $(AG)[P \rightarrow -R]$—if a student has passed a course, then s/he is not registered for it.

(6) $(AG)(R \rightarrow -P)$—if a student is registered for a course, then s/he has not passed it.

Suppose the question were posed "Is John registered for CSE110?". By virtue of the above axioms, there are three possibilities, depending on what facts are in the data base.

Possibility 1. John is not registered for CSE100 ($\bar{}R$) but has passed it (P).

In this case, the direct answer would be "No, he is not registered for CSE110." If we now were to consider *monitoring* for the event "John registers for CSE110", we could rule it out on the basis that it is provable that there can be no future in which John is registered for CSE110. Specifically, from P and axioms 1 and 5, it is provable that $-(EF)R$. It would therefore be incompetent to offer to monitor for that condition.

Possibility 2. John neither has passed nor is registered for CSE110 ($-P \ \& \ -R$).

In this case, the direct answer is again "No". However, we could offer to monitor for John registering for CSE110, since $(EF)R$ is derivable from axiom 2, i.e. there is some future in which, at some point, John is registered for CSE110.

Possibility 3. John is registered for CSE110 (R).

In this case, the direct answer is "Yes". Moreover, we could competently offer to monitor for any of the following additional events:

(a) John no longer registered for CSE110; $(EF)-R$.

(b) John passed CSE110; $(EF)P$.

(c) John registered for CSE110 again; $(EF)[-R \ \& \ (EX)R]$.

This last case (3c) is interesting in that it can be viewed as a monitor for $-R$, whose action is to set a monitor for R (whose action is to inform the user of R).

Work in this area is still in an early stage. We are currently using a straight-forward Tableau Method theorem prover which is able to reason about possible events or states in this branching-time temporal logic. Improving its efficiency and extending its coverage are among our current areas of research.

(2) *Revelance.* The relevance of a monitor offer depends on whether or not the information, upon arrival, would still be of interest to the user. Because the information differs, we distinguish the discourse functions that monitor offers support: (1) that of deferring a response, when the system cannot answer the user's query immediately, and (2) that of providing additional information that the user has not explicitly requested.

In the first case, the system must determine whether the user would still be interested in getting the explicitly requested information from the system sometime later. This depends in part on why the system has to defer its response: that is, either the information about an actual event hasn't been registered in the system yet *or* the event itself hasn't yet happened. In the first case, the user might be able to get the information elsewhere, but not in the second. Therefore, the system must determine what role the information fills with respect to the user's goals—what problem s/he is trying to solve and what time constraints s/he is under in doing so. These form the aspects of a user model that are of significance here.

The second case is somewhat different and can be divided into two subcases, depending on what additional information the system offers to monitor for. If we look at a query as asking whether a given description once held, then one sort of additional information the system can offer is whether that same description will hold at some future time (whether or not it has already held in the past). For example,

U: Has John taken CIS531 yet?
S: No. Shall I let you know if he does?
U: Did John visit NSF last week?
S: Yes. Shall I let you know if he does so again?

Alternatively, the system can offer to monitor for whether some related description will hold at some future time. For example,

U: Did John check into the Statler Monday?
S1: Yes. Shall I let you know when he checks out?
or
S2: Yes. Shall I let you know when the rest of the committee does?

Again, whether such information would be of interest to the user requires a model of the user's goals, but our feeling is that in this second case, it must be his/her more long-range, global informational goals. Moreover, we probably only want to concern ourselves with very sharply defined, strongly held goals, in order to restrict which "somehow related" descriptions would be considered.

Another type of monitor that may be of interest to the user, in a different way, is the complement of those so far discussed. That is, the user may be interested in learning that some earlier set monitor can no longer be satisfied, so that s/he does not continue to expect it to go off. For example, if the system has offered to inform the user if John registers for CSE110, then s/he should be informed if John receives advanced placement credit and can no longer register.

Finally, it may be the case that it is more appropriate to take some other type of initiative than offering to monitor (S1), perhaps justifying to the user what the system's *current* answer follows from (S2), as we shall discuss in the next section.

U: Did anyone fail CIS531?
S1: Yes, John and Martha. Shall I let you know if anyone else does?
S2: Yes, John received a C and Martha a D. Graduate courses require a B as a passing grade.

As noted before, work in this area is still at an early stage, especially with respect to a model of the user's informational goals, and we hope to have more specific results as we go along.

3(c) *Justification*

The third type of initiative-taking behavior to be discussed involves the system's *justifying* what its response follows from. We strongly believe that a user should not just have to accept the system's answer without comment. Rather, s/he is entitled to be given an *informed basis* for either accepting it or questioning the premises or reasoning involved in its derivation. In this regard, justification differs from *argument*, whose sole role is to *convince* a listener to do or believe something. Giving the user the means to make informed judgments is our motivating force behind this work.

There are at least three important issues involved in producing justifications:

(1) What justification does a query call for?
(2) What justification can the system give?
(3) What constitutes a clear and understandable justification?

(1) *What justification does a query call for*? This depends primarily on whether or not the user's perceived expectations have been fulfilled by the system's answer. For example, the use of "still" in a query indicates that the user expects that the potentially changing situation described in his/her query has not in fact changed. For example,

U: Does anyone still have an Incomplete in CSE110?
S1: Yes, John does.
S2: No. The last remaining Incomplete was made up on 1 December.

If the user's expectation is correct (i.e. not all Incompletes have been made up), then facts corroborating that expectation may constitute appropriate justification, as in S1. (A monitor offer may also be appropriate, as in the "yet" case noted earlier.) If it isn't correct (as in S2), then specifying the event that enabled/caused the change may constitute a more appropriate justification.

More often than not, the user's expectations will not be clearly identifiable. Again, a user model is called for, but here the significant aspects involve a characterization of what situation the user believes to hold and what s/he expects to learn from his or her queries. As noted earlier, this is the focal point of much research.

(2) *What justification can the system give*? In the case where the system knows no more than the particular facts in its data base, the only justification it can provide is those facts themselves (or some description of them). For example, suppose the system knows for each student-course pair only whether the student has passed or failed the course. Then only the following kind of justification is possible:

U: Did anyone fail CIS531?
S1: Yes, John and Martha.
S2: Yes, 50 seniors and 12 graduate students.

On the other hand, some reasoning may have been involved in deriving the answer, as in the case where the system knows (e.g. has an axiom to the effect) that "failing" a graduate course follows from a receiving a grade of C or worse. In this case, specifying the particular grade each failing student received could be used to justify the system's answer, i.e.

U: Did anyone fail CIS531?
S: Yes, John received a C and Martha, a D.

Or the system can also give some indication of its reasoning as well, as in

U: Did anyone fail CIS531?
S: Yes. With a graduate course, everyone fails who gets a C or below. In CIS531, John received a C and Martha, a D.

We believe that possible justifications are *related* but not *equivalent* to how the system comes to know an answer. The reason for this reservation has to do with the third issue mentioned earlier in justifying a direct answer, which we shall discuss next.

(3) *What constitutes a clear and understandable justification*? Answers to data base queries may be derived by straight look-up, by computation and/or by reasoning. Where reasoning is involved, producing clear and understandable justifications involves notions of (1) *succinct* justifications that do not say *more* than is necessary (lest they lose the user) and (2) conceptually *well-structured* justifications that the user can understand and evaluate.

There are many techniques for conveying information succinctly. The one we shall discuss here for illustration is a local technique for abbreviating proofs, that involves conveying part of one's reasoning by implication and has been called by Sadock[21] *modus brevis*. The notion of well-structured justifications we will discuss in terms of high-level strategies for reasoning. These highlight the strong similarity we see between reasoning and hierarchical planning.

(i) *Succinct justifications—modus brevis*. As a simple illustration of *modus brevis* and its use in making justifications *succinct*, consider a *modus ponens* deduction, possibly used in deriving an assertion and now to be used in justifying it. It has been observed that in presenting such reasoning one need not make explicit all three parts of the proof—the major premise ($A \rightarrow B$), the minor premise (A), and the conclusion (B). Rather it is sufficient to state the conclusion, with either the major premise or the minor premise (but not both) as support. The listener is assumed able to infer the connection between the two—that is, they are steps in a proof that requires the missing premise to complete.

So suppose in response to the query "Did John fail physics?", the system makes the following *modus ponens* deduction:

```
Anyone who gets below a C fails physics. (* major premise *)
John got below a C. (* minor premise *)
John failed physics. (* conclusion *)
```

The system can then justify its "yes" answer in either of the following ways, relying on the user's ability to recognize the underlying deduction.

S: Yes. Everyone failed physics who got below a C.
S: Yes. He got below a C.

(Why the speaker chooses to give major premise over minor, or vice versa, is a matter yet to be investigated.)

Modus brevis can also be used in presenting other types of reasoning succinctly, both deductive and non-monotonic[22]—for example, "typicality" reasoning:

U: Is John taking Physics 281?
S1: I don't know, but probably. Students typically take Physics 281 after Physics 280. (* conclusion and major premise *)
S2: I don't know, but probably. He took Physics 280 last term. (* conclusion and minor premise *)

In any case, the speaker must be able to assume that the listener can, on the basis of what is essentially a *clue* to an argument, reconstruct that argument. Whether the listener is *convinced* by the argument s/he deduces, i.e. whether s/he *accepts* the inferred premise, is a separate issue: the listener can always attempt to confirm that s/he has inferred what the speaker has intended or to challenge it. For example,

U: Did John fail physics?
S: Yes. He got a B.
U: Is the failing grade really B or below?

(This goes beyond what has standardly been taken to be a "clarification dialogue" in Natural Language systems. The function of such a dialogue is usually seen as identifying terms and phrases which haven't been understood by the listener, or which are ambiguous, etc.[23].)

Since the successful use of *modus brevis* in justifications depends on the listener's ability to recognize the relevance of the additional material in the system's response as being part of some (implied) chain of reasoning, its primary use will be in place of very short reasoning chains, rather than in justifying an entire (complex) proof. Most of the techniques for producing succinct justifications will probably also have this local flavor.

(ii) *Well-structured justifications—hierarchical reasoning.* We believe there are appropriate schemas for presenting reasoning that are essentially independent of content. These schemas correspond to *valid* reasoning strategies (some of which are noted below), that everyday reasoning aspires to. That is, people usually explain themselves so as to give the impression that their reasoning is valid. Such explanations, while rarely adhering religiously to valid methods, are nevertheless understandable because (1) the strategies are accepted and (2) textual devices like "on the other hand", "suppose ⟨*x*⟩ were true", etc. make clear what strategies are being used.

Proofs using these reasoning strategies (and correspondingly, the justifications and explanations that follow them) bear a strong resemblance to structures found in hierarchical planning. This is not strange if one views reasoning as actions taken to support or deny a proposition. Just as hierarchical strategies for actions can be used in *forming* plans, *revising* plans, or *describing* them to another person[10, 24], so hierarchical reasoning strategies can be used in *constructing* a proof or *justifying* a result.

What reasoning strategies are we talking about? Many researchers have already observed that explanations have a tree-like structure, in which supported assertions correspond to non-terminal nodes with their support making up the sub-tree under them[25, 26]. Since a statement acting as a reason may in turn be supported by other statements/reasons, explanations have a recursive or hierarchical structure. This is *not* what we are talking about. The kinds of hierarchical reasoning strategies we have in mind are things like:

- *Simple case analysis*—to show that Q is true, find some proposition P from which Q from whose simultaneous satisfaction Q follows. For each Pi, show that it follows. Hence Q must be true.
- *Simple case analysis*—to show that Q is true, find some proposition P from which Q

follows, independent of P's truth value. Assume P and show that Q follows. Assume $\neg P$ and show the same. Since either P or $\neg P$ must be true, Q must be true. (Alternatively, to show Q is false, find some P from which $\neg Q$ follows, independent of P's truth value. Assume P and show $\neg Q$ follows. Do the same for $\neg P$. Since P or $\neg P$, $\neg Q$ must be true—hence Q is false.)

- *General case analysis*—to show that Q is true, find some assertion P that is partitionable into $P1, \ldots Pk$. Assume each Pi in turn and show that Q follows from Pi. Since some Pi must be true given P is, Q must be true. (This has the obvious complementary strategy for showing Q false.)
- *Reduction* ad absurdum—to show that Q is false, find some proposition P whose both assertion and negation follow from Q. Assume Q and show that P follows. Show that $\neg P$ follows. Since Q leads to both P and $\neg P$, Q must be false.

(Other strategies are described in [18].)† These strategies are hierarchical or recursive in that wherever a strategy calls for showing "P follows" or "$\neg P$ follows", there another strategy may be chosen and invoked in support. That such strategies can be used in reasoning is well-known. What is significant here is that where a justification is organized according to such strategies, it is that much easier to follow.

To illustrate this, consider the following tale, whose humor follows in part from the recursive use of *simple case analysis* in support of successive alternatives.

"WHAT IS THERE TO BE FRIGHTENED OF?"

War was on the horizon. Two students in the Yeshiva were discussing the situation.

"I hope I'm not called," said one. "I'm not the type for war. I have the courage of the spirit, but nevertheless I shrink from it."

"But what is there to be frightened about?" asked the other. "Let's analyze it. After all, there are two possibilities: either war will break out or it won't. If it doesn't, there's no cause for alarm. If it does, there are two possibilities: either they take you or they don't take you. If they don't, alarm is needless. And even if they do, there are two possibilities: either you're given combat duty, or non-combatant duty. If non-combatant, what is there to be worried about? And if combat duty, there are two possibilities: you'll be wounded, or you won't. Now if you're not wounded, you can forget your fears. But even if you are wounded, there are two possibilities: either you're wounded gravely or you're wounded slightly. If you're wounded slightly, your fear is nonsensical, and if you're wounded gravely, there are still two possibilities: either you succumb and die, or you don't succumb and you live. If you don't die, things are fine, and even if you do die, there are two possibilities: either you will be buried in a Jewish cemetery or you won't. Now if you're buried in a Jewish cemetery, what is there to worry about, and even if you are not... but why be afraid? There may not be any war at all!"[27].

In this example, "there's no call for worry" is the Q meant to be proven. The initial P being used to support Q independent of its truth value is "war will break out". Assuming $\neg P$ (i.e. war won't break out), then Q follows because the derivable major premise $\neg P \rightarrow Q$ is accepted as true. (This is an instance of *modus brevis*.) On the other hand, to show Q follows from assuming P, the speaker invokes a simple case analysis strategy again, this time finding P'—"they take you [into the army]"—meant to support Q independent of its truth value. Assuming $\neg P'$ (i.e. they don't take you), then Q follows because the derivable major premise $\neg P' \rightarrow Q$ is again accepted as true. On the other hand, to show Q follows from assuming P', the speaker invokes simple case analysis again, finding a P'', etc.

The point we want to make is that whereas the reasoning involved in finding a proof should be as computationally efficient as possible, that proof may not itself be structured in such a way as to map directly onto an understandable justification. Rather we want a justification that conforms to some "reasoning plan" using strategies such as those above, that is *faithful* to its derivation but not necessarily a mirror of it. The question is whether one could take a proof,

†"Follows from" is more than "implies": it represents the proof relation ($\vdash$), and not simply the implication connective ($\rightarrow$). While the above strategies refer to this proof relation, we believe there are comparable strategies for the more general notion of "gathering support" used in commonsense, probabilistic and other types of non-deductive reasoning.

recognize from it which of the above understandable reasoning strategies could be used in justifying the result, and then *construct* an appropriate valid justification in terms of those strategies. We discuss this issue in greater detail in [18], but research is still only in its early stages.

4. CONCLUSION

In this paper we have attempted to describe additional features that could usefully be incorporated into a Natural Language system for interfacing with data bases. Such extensions are not a luxury: they are necessary if today's data base question-answereing systems are to become tomorrow's decision support systems. We have argued elsewhere[28] that such systems, if they are to satisfy the legitimate needs of their users, must include communicative capabilities as sophisticated as those that are the subject of current Natural Language research. This paper has described several of those capabilities.

REFERENCES

1. W. Woods, R. Kaplan and B. Nash-Webber, The lunar sciences natural language information system: Final report. *Tech. Rep.* 2378. Bolt, Beranek & Newman, Cambridge, Mass. (1972).
2. G. Hendrix, E. Sacerdoti, D. Sagalowicz and J. Slocum, Developing a natural language interface to complex data. *ACM Trans. Database Systems* 3(2), 105–147 (1978).
3. D. L. Waltz, An English language question answering system for a large relational database. *Commun. ACM* 21(7) (1978).
4. S. Kwasny and N. Sondheimer, Relaxation techniques for parsing ill-formed input. *Am. J. Comp. Ling.* 7(2), 99–108 (1981).
5. A. Malhotra, Design criteria for a knowledge-based English language system for management: an experimental analysis. *MAC TR*-146, MIT, Cambridge, Mass. (1975).
6. H. Tennant, Experience with the evaluation of natural language question answerers. Working paper #18, Univ. of Illinois, Urbana-Champaign, Ill. (1979).
7. K. McKeown, Generating natural language text in response to questions about database structures. *Tech. Rep. CIS*-82-5, Dept. of Computer and Information Science, University of Pennsylvania (May 1982).
8. J. E. Grimes, *The Thread of Discourse.* Mouton, The Hague (1975).
9. K. F. McCoy, Augmenting a database knowledge representation for natural language generation. *Proc. 20th Ann. Meeting Assoc. for Comp. Linguistics*, Toronto, Canada (June 1982).
10. B. Grosz, The representation and use of focus in dialogue understanding. *Tech. Rep.* 151, SRI International, Menlo Park, CA (1977).
11. C. L. Sidner, Towards a computational theory of definite anaphora comprehension in English discourse. Ph.D. dissertation, MIT, Cambridge, Mass. (1979).
12. S. Bossie, The tactical component for text generation: sentence generation using functional grammar. Forthcoming Masters thesis. University of Pennsylvania.
13. S. J. Kaplan, Cooperative responses from a portable natural language data base query system. Ph.D. thesis. Department of Computer and Information Science, University of Penn. (1978).
14. E. Mays, Correcting misconceptions about data base structure. *Proc. 3-CSCSI*, Victoria, B.C. (May 1980).
15. E. Mays, S. Lanka, A. K. Joshi and B. L. Webber, Natural language interaction with dynamic knowledge bases: Monitoring as response. *Proc. 8-IJCAI*, Vancouver, B.C. (August 1981).
16. E. Mays, A. K. Joshi and B. L. Webber, Taking the initiative in natural language data base interactions: monitoring as response. *Proc.* 1982 *European Conf. Artificial Intelligence*, Orsay, France (July 1982).
17. E. Mays, Monitors as Responses to questions: determining competence. *Proc.* 1982 *Natl Conf. Artificial Intelligence*, Pittsburgh, Penn. (August 1982).
18. B. Webber and A. Joshi, Taking the initiative in natural language data base interactions: justifying why. *Tech. Rep. MS-CIS*-82-1, Dept. of Computer and Information Science, University of Pennsylvania, April 1982. (A shorter version of this paper appears in *Proc. Coling*-82, Prague, Czechoslovakia, July 1982.)
19. A. K. Joshi, Mutual beliefs in question-answer system. In *Mutual Belief* (Edited by N. V. Smith). Academic Press, New York (1982).
20. M. Ben Ari, Z. Manna and A. Pneuli, The temporal logic of branching time. *8th Ann. ACM Symp. Principles of Programming Languages*, Williamsburg, Virginia, January 1981.
21. J. Sadock, Modus brevis: the truncated argument. Papers from the *13th Regional Meeting Chicago Linguistics Society*, Chicago, Ill, 1977.
22. *Artificial Intell. J.*, Special issue on Non-monotonic Logic 13(1) (1980).
23. E. F. Codd, Seven steps to rendezvous with the casual user. In *Data Base Management* (Edited by J. W. Klimbie and K. Koffeman). North-Holland, Amsterdam (1977).
24. E. Sacerdoti, *A Structure for Plans and Behavior.* Elsevier, New York (1977).
25. J. Weiner, BLAH, a system which explains its reasoning. *Artificial Intell. J.* **15**, 19–48 (1980).
26. R. Cohen, Investigation of processing strategies for the structural analysis of arguments. *Proc. 19th Ann. Meeting of the Assoc. for Computational Linguistics*, Stanford, Calif., June 1981.
27. N. Ausubel (Ed.), *A Treasury of Jewish Folklore.* Crown, New York (1948). (Abridged edition published by Bantam, 1980.)
28. M. Pollack, J. Hirschberg and B. Webber, User participation in the reasoning processes of expert systems. *Proc.* 1982 *Natl Conf. Artificial Intelligence*, Pittsburgh, Penn., August 1982.