stored in the data base, or have a random number generator select one randomly. But, if more than one item is in stock, it is most probable that the user would like to see some of the other items as well. In fact, he probably wants to see all the items if the list is not "too long." It is not clear at all how the user is expected to ask for more items after the response to (1). Is he expected to make a new request, perhaps phrased as in (2)? Or is he expected to retype the same request as in (1) again, and to obtain a different item? Under the first item selection, the same (1) request will produce the same results. Hence, there is no way for him to get a different item printed. Under the random item selection, he would probably get a different item each time the same (1) request is executed; however, he has no control over which item is printed. Without user control it is doubtful that any selection strategy is useful.

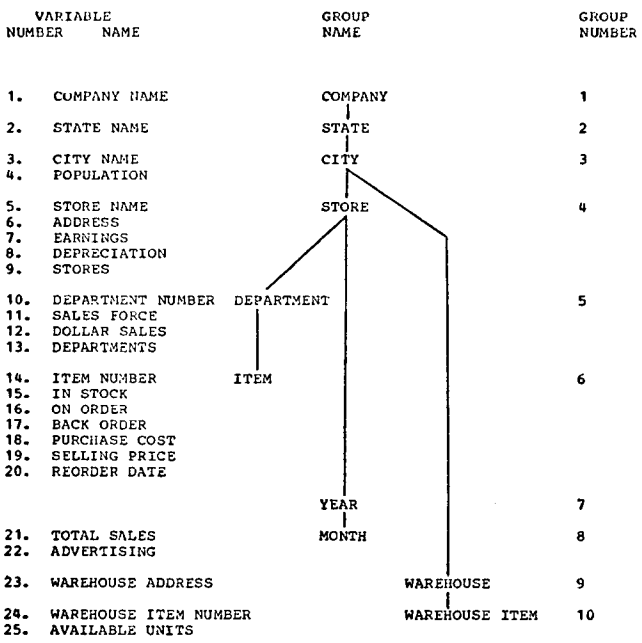

Figure 1 Hierarchical structure of an arbitrary inventory data base.

Thus the detectable syntactic difference between (1) and (2) is nullified from the point of view of the users. Furthermore, if this syntactic difference in number agreement is to be consistently applied, it is not clear what the system should do when this agreement is not observed. For example, Montgomery* (p. 1077) cites "real" requests from nuclear physicists that violate this number agreement, and yet the meanings of the requests are perfectly clear:

(3) Does experimental data show whether antiparticles have negative initial mass?

(4) In the measurement of single spectra Beta radiation, how must the absorptive material be placed ...?

Similarly, there are many other syntactically different requests that are the same to the user.

(5) What are the stock items?
(6) What are the items which are in stock?
(7) What stock items are there?
(8) What stock items are available?

If the information management system is to accept English requests, it must somehow conclude that requests (5) through (8) (and many other different constructions) are identical. Yet at the same time, the system must judge that requests (9) through (12) are somehow all different from those shown in requests (5) through (8):

(9) What stock items are unavailable?
(10) What items are out of stock?
(11) Which stock items have all sold out?
(12) List all the stock items which must be reordered immediately.

2.2 Problems of Semantics

Conjunction is one of the most powerful productive processes for forming new sentences in English from two or more related sentences. Chomsky[5] illustrates the conditions which allow the conjunction to take place. For example, he states (p. 25) that it is possible to form (14) from (13a) and (13b):

(13) (a) The scene -- of the movie -- was in Chicago.
(b) The scene -- of the play -- was in Chicago.
(14) The scene -- of the movie and of the play -- was in Chicago.

But it is not possible to form (16) from (15a) and (15b):

(15) (a) The -- liner sailed down the -- river.
(b) The -- tugboat chugged up the -- river.
(16) The -- liner sailed down the and tugboat chugged up the -- river.

Chomsky then proposes that in order to conjoin two sentences, at least the following requirement must be satisfied:

> "If S_1 and S_2 are grammatical sentences, and S_1 differs from S_2 only in that X appears in S_1 where Y appears in S_2 (ie, S_1 = ...X...and S_2

= ...Y...), and X and Y are constituents of the same type in S_1 and S_2 respectively, then S_3 is a sentence, where S_3 is the result of replacing X by X + and + Y in S_1 (ie, S_3 = ...X + and + Y ...)."

For an information management system using English, one must be able to allow conjunction constructions. Hence, if the user wants to execute two requests, as in (17) he should also be able to execute a request like (18):

(17) (a) Which store has the highest dollar sales?
(b) Which department has the highest dollar sales?
(18) Which store and department have the highest dollar sales?

Indeed, (18) appears to be a paraphrase of the two sentences in (17). However, when there exists a hierarchical relationship between the store and the department as that shown in Figure 1, then the interpretation of (18) may not be the same as the two sentences in (17). (17a) says that among all the stores, print that one which has the highest dollar sales. (17b) says that among all the departments (in one store or many stores combined), print that department which has the highest dollar sales. For (18) it is not clear which one of (19) the user intends.

(19a) Among all the stores, print that store which has the highest dollar sales; then for that store, print the department which has the highest dollar sales.

(19b) Among all the departments, print that department which has the highest dollar sales; then print the store which contains that department.

(19c) Among all the stores, print that store which has the highest dollar sales; then for all the departments, print that department which has the highest dollar sales.

The interpretations (19a) and (19b) preserve the hierarchical relationship, that is, the printed department name belongs to the printed store name. The interpretation (19c) does not in general print a department that belongs to the printed store; however it does produce the same results as the two separate requests in (17).

The above examples of using conjunctions illustrate the potential ambiguity in combining two sentences with two different subjects. A different kind of problem arises when the objects of the two sentences are different. For example, (21) is a correct conjunctive construction from (20a) and (20b) in accordance with Chomsky's minimal syntactic requirement given above.

(20) (a) Which store has the highest earnings?
(b) Which store has the lowest earnings?
(21) Which stores have the highest earnings and the lowest earnings?

One interpretation of (21) is that it is equivalent to the two requests shown in (20). There is a second possible interpretation, however. That interpretation considers the two objects as a logical relation using AND. That is, each store must satisfy the condition,

(earnings is highest) and (earnings is lowest)

This interpretation might be rejected on semantic grounds, since it is generally not likely that a store has both the highest and the lowest earnings at the same time. However, if the two objective noun phrases do not use the same noun, the interpretations are not that clear anymore. For example,

(22) Which stores have the highest earnings and lowest depreciation?
(23) Which items have purchase cost less than $10 and selling cost greater than $20?
(24) Which items have less than 100 in stock and were reordered during May, 1972?

For all of these three examples, it is more likely that the interpretation is one of logical AND rather than two separate requests. One might then conclude that whenever the two objective noun phrases have the same noun, then the interpretation is one of separate requests; while if the two objective noun phrases have different nouns, then its interpretation is one of logical conditions. Unfortunately, this conclusion is not true, since the interpretation for (25) is clearly one of logical condition:

(25) Which stores have greater than $100,000 earnings and less than $200,000 earnings?

To complicate the matter, the use of OR can be very misleading too, as in,

(26) Which stores have the highest earnings or lowest depreciation?

This request could mean,

(27) (a) Which store has the highest earnings?

or

(b) Which store has the lowest depreciation?

Under such an interpretation, one could randomly print one of the two results (which would be a bad human factors design, since the responses are unpredictable), or what is more usual, to print both results. In the latter case, the answers are exactly equal to (22) when the conjunction AND is taken to mean,

(28) (a) Which store has the highest earnings?

and

(b) Which store has the lowest depreciation?

Alternatively, request (26) could mean that each store must satisfy the logical condition of,

(earnings is highest) or (depreciation is lowest)

Either interpretation of (26) produces the same set of stores. Consequently the conjunctive-use of OR is equivalent to the logical-use of OR. However, the conjunctive-use of AND is definitely not equivalent to the logical-use of AND. There is no indication from the English language usage as to which interpretation of AND the user intends when he uses the AND conjunction.

Kellog[6] states well the problems of natural language communication with the computer (p. 475):

> "The heart of the difficulty in dealing with natural language for the computer is not only the great complexity of the syntactic structures which may arise in a natural language, but even more so, the complexity of the associations and relations which may obtain between members of syntactic categories and members of semantic categories."

(Incidentally, I would challenge any existing or future natural language analyzer to correctly parse and interpret the quoted sentence above consisting of 54 words, 2 commas and a period.)

Other language ambiguities which result from the use of conjunctions are given by Smith[7] and Gleitman[8].

2.3 Problems of Typing

A typical user of an information management system is not an expert typist. He tends to make typing errors quite often. It is too presumptuous for any information management system to assume that all requests are correctly typed. In fact, one of the most difficult tasks for an information management system is to detect typing errors.

A mistyped word is not any different from a correctly typed word which the information management system cannot recognize. Hence the ability to detect typing errors is tied to the overall procedure in the lexical analysis of the input request. This procedure includes at least a dictionary lookup of some kind. If the dictionary lookup fails, then either the word does not belong to the list of acceptable lexical entries or else it is erroneously typed. The SMART[9] system, for example, ignores all words that are not found in the dictionary lookup. Under such circumstances there is no way to detect typing errors. It is then highly probable that a misspelled word (such as, no and not) could misrepresent the request entirely.

Most information management systems which use English reject the request whenever any unrecognized, lexical entry is detected. The user must then retype the entire request. For example, Winograd's[10] system interrupts the typing whenever an unrecognized word is detected. It types out the message,

> SORRY, I DON'T KNOW THE WORD "X".
> PLEASE TYPE TWO SPACES.

The two spaces are necessary to clear internal buffers. Whenever the REL[11] system of Caltech cannot recognize a request, it types out a most uninformative message,

> Eh?

For example,

> What is the sex ratio of total 1970 Mazulu sample?

causes the error message,

> Eh?

The error, believe it or not, lies in the simple singular versus plural form of "sample," that is, the request should have been typed as,

> What is the sex ratio of total 1970 Mazulu samples?

This kind of overdependence on inflectional endings is unrealistic, especially in view of the fact that the following request was accepted.

> What is the number of total 1957 Mazulu sample who are male?

It is one thing for an information management system to detect typing errors. It is much more difficult to provide short and concise diagnostic messages to inform the user about what to correct.

It is highly desirable for an information management system to allow error corrections on a word basis, rather than on the entire request; otherwise the information management system is biased against the use of long requests. For a fairly long request, say two lines, it is too much to require the user to retype the entire two-line request when the only error was a permutation of two letters in a word.

III. COMMUNICATING IN KEYWORDS

The above section dealt with the problem of "understanding" requests expressed in English. Under certain restrictive environments, it is possible to communicate with an information management system in a well defined subset of the English language. But what that subset is, is generally not known to the user who is making his retrieval request. In fact, as Senko[2] comments (p.247),

> "It is almost certain[ly] more difficult to learn the characteristics of this subset [of English] than it is to learn the characteristics of the subset of 'English-like Boolean languages.'"

In place of English, the user can be taught to use the special language for communicating with an information management system. This language should be easy and natural for users to learn and to use effectively, and yet be very precise so that there is minimal ambiguity in formulating requests. The Natural Dialogue System[12] (NDS) is a vehicle for implementing a class of languages based on a keyword style of dialogue first described by N. R. Sinowitz[13]. The history and philosophy behind the development of such an interactive system is covered by Wier[14]. A specific query language, implemented in the Off-the-Shelf System, is described by by Heindel and Roberto[15]. Some of the features of the CISS language will be used to illustrate its capabilities as a query language for an information management system.

The keyword style of dialogue is a means of communicating with an information management system in which a request is broken down into a number of independent statements or phrases. Each begins with a keyword and terminates with a colon (:). The keyword specifies the kind of function to be performed while the parameters following the keyword specify the data upon which the function is to be performed. This style of dialogue appears strange at first since it does not follow the predominant subject-verb-object pattern of English sentence construction. Instead, each statement begins with a keyword which indicates its function in a request. For example, verbs such as PRINT, RANK, and DISTRIBUTE indicate specific actions upon the retrieval data; modifiers such as IN, FOR, and WHEN specify subsets of the data base for the current request; and auxiliary keywords such as FREE, RECAP, and TIME are user-aids to facilitate the dialogue.

The keyword dialogue has many advantages, some of which are the following:

1. It reduces ambiguities.
2. It allows the user to think in smaller functional units.
3. It allows the user to selectively edit part of his request (one phrase at a time) without destroying the rest.
4. It forces the user to be clear.
5. It takes away the incentive to "play" against the system to discover novel sentences that the system cannot handle.

This keyword style of dialogue relies heavily on the user to provide unambiguous requests. This is based on the assumption that only the user truly knows what data are needed in his current request. It is worse to print results based on an erroneous interpretation than it is not to print at all. The potential danger of making a critical decision based on an incorrect interpretation is considerable.

3.1 Forming Requests in OTSS

In this section, some of the English examples presented earlier in this paper are now given in the keyword style used in the OTSS. Six of the first eight requests are all equivalent to each other. The OTSS equivalent is the following,

```
(29) = (1,2,5 thru 8) PRINT ITEM NUMBER:
                       WHEN IN STOCK>0:
```

Notice that two statements are required to express the request. This is because the phrase "items which are in stock" refers to two separate data variables; one to give the item number and the other to give the amount of that item which is in stock. Consequently, those items that are currently in stock are precisely those whose IN STOCK values are positive. If this request produces a voluminous output, the user can easily limit the request to one or more departments, stores, cities, or states by including a modifier such as,

```
FOR BUFFALO; SYRACUSE:
```

This modifier delimits the data base to the cities of Buffalo and Syracuse while retaining the original PRINT and WHEN statements.

The ability to modify part of a request and the relative ease with which this may be done results in a very flexible dialogue between the information management system and the users.

For the two requests shown in (17), the equivalent requests are the following:

```
(30) = (17a)  LET X = $ SALES PER STORE:
              PRINT STORE NAME:
              WHEN STORE HAS X =
               MAX X PER COMPANY:

(31) = (17b)  PRINT DEPT NAME:
              WHEN DEPT HAS $ SALES =
               MAX $ SALES PER COMPANY:
```

The reason for creating a variable X is to obtain the DOLLAR SALES (or $ SALES) for each store. The WHEN statement in (30) restricts the printouts to the store(s) for which X is maximum among all those in the entire company. The WHEN statement in (31), on the other hand, restricts the printouts to the department(s) for which $ SALES is maximum among all those in the entire company. In this keyword style of language, it is up to the user to specify the group (whether it be the STORE or the DEPT) upon which the maximization is to be performed.

The conjunctive request shown in (18) has three separate interpretations as shown in (19a) through (19c). These three interpretations can be precisely expressed as shown below, which the reader can verify after reading reference 15 on the CISS system.

```
(32) = (19a)  LET X = GLOBAL $ SALES
               PER STORE:
              PRINT STORE NAME,
               DEPT NAME:
              WHEN STORE HAS X =
               GLOBAL MAX X PER COMPANY:
              WHEN DEPT HAS $ SALES =
               MAX $ SALES PER STORE:

(33) = (19b)  LET TORF = $ SALES =
               GLOBAL MAX $ SALES
               PER COMPANY:
              PRINT STORE NAME,
               DEPT NAME:
              WHEN STORE HAS ANY TORF
               PER STORE = TRUE:
              WHEN DEPT HAS TORF = TRUE:

(34) = (19c)  LET X = $ SALES PER STORE:
              LET Y = IF X = MAX X
               PER COMPANY,
               THEN STORE NAME,
               ELSE AN:
              LET Z = IF $ SALES = MAX $
               SALES PER COMPANY,
               THEN DEPT NAME,
               ELSE AN:
              PRINT Y,Z:
```

The LET statement in (32) creates a variable X which is simply the $ SALES for a store. The use of GLOBAL is a modifier which overrides the selection conditions at the group of the defining variable (which is $ SALES at the DEPT group) and all the groups up to the group in which the variable is raised (in this case, STORE). Hence the effect of GLOBAL is to ignore the WHEN statement at the DEPT group. The first WHEN statement in (32) selects the one store which has maximum X (i.e. $ SALES for a store) among all the stores in the company. The effect of the literal GLOBAL in this WHEN statement is to ignore the WHEN statement at the STORE group. The second WHEN statement in (32) selects the one department whose $ SALES is maximum among all the departments in the previously selected store.

In (33), the LET statement creates a logical variable TORF (read "t or f") which will have the value TRUE for the department(s) having the maximum $ SALES among all the departments in the company; and a value FALSE otherwise. Again, the use of GLOBAL is to override the selection conditions at the DEPT and at the STORE groups. The first WHEN statement selects the store(s) in which at least one department has the value TRUE for TORF. The second WHEN statement simply selects that department within the selected store for which TORF is TRUE. Hence the selected store contains the selected department, as required by interpretation (15b).

In (34), we need to create two separate variables Y and Z which are, independently, maximum $ SALES at the store and at the department groups, respectively. We need to ignore the printing of these values when they are not the maximum values. Hence the IF-THEN-ELSE construction is used to assign AN (absent node) to those non-maximum values. The effect of AN on the PRINT processor is simply to ignore it.

When the conjunction AND is used to conjoin two objective noun phrases as in (21), the interpretations are not clear. The two possible interpretations for (21)

can be precisely stated in the keyword style as shown in the following,

```
(35) = (21)   PRINT STORE NAME:
              WHEN EARNINGS = MAX
               EARNINGS PER COMPANY
               AND EARNINGS = MIN
               EARNINGS PER COMPANY:

(36) = (21)   PRINT STORE NAME:
              WHEN EARNINGS = MAX
               EARNINGS PER COMPANY
               OR EARNINGS = MIN
               EARNINGS PER COMPANY:
```

Notice that in (36), the conjunction OR is used to express an English request which uses the conjunction AND. Similarly, dual keyword requests may be formulated for all the ambiguous English requests (22) through (25).

As for (26) where the conjunction OR is used, the two interpretations produce the same stores; hence its equivalent keyword form is,

```
(37) = (26)   PRINT STORE NAME:
              WHEN EARNINGS = MAX
               EARNINGS PER COMPANY
               OR DEPRECIATION = MIN DE-
               PRECIATION PER COMPANY:
```

It is clear from the above sample requests that OTSS provides a way to formulate precise requests, although in some cases, the formulations are not simple.

3.2 Independence of Statements

Each statement in a keyword style language performs certain functions. A request is made up of one verb with as many of its modifiers as are active when executing the request. The modifiers generally delimit the data base to a subset by either specifying a section of the hierarchy (through IN and FOR keywords) or comparing data values (through WHEN). Other modifiers are used to control the format of printout, to provide titles, and for creating additional variables which are arbitrary functions of other variables.

The order in which the modifiers are given is immaterial, except that all newly created variables must be defined before they are used in any other statement. Successive requests can thus be given by retyping one or more statements, instead of the entire request. This editing feature, which is based on individual statements, gives the user a very flexible way of successive modification of his requests to the information management system. Most importantly, all extraneous typing of unchanged statements is avoided. The editing of a hierarchical modifier IN, for example, allows the user to see the effect of the same request for a different location; while the editing of the verb allows the user to see the effect of the same data being manipulated and presented differently through, say, DISTRIBUTE, RANK or STATISTICS.

IV. SUMMARY

This paper presents a number of arguments against using English as the query language for information management systems. In place of English, a style of query language based on keyword dialogue is presented by giving examples for the equivalent English sentences. The underlying assumption for the query language is that only the user knows what information he wants. Hence the language must provide the vehicle for the user to express precisely a large class of requests.

It should be pointed out that OTSS is only one example of a query language using NDS to interface with the user and using Master Links[16] to access the data base. Other query languages can be specially tailor-made for a particular community of users using the tools provided by NDS and Master Links. The style of the query language will remain that of keyword dialogue, but the syntax and the semantics of the new query language can be completely specified by the designer of the query language. An example of such a special tailor-made query language was presented elsewhere by Chai[17].

REFERENCES

1. C. A. Cuadra (editor), Annual Review of Information Science and Technology, (Chapters on Automated Language Processing), Interscience Publishers, 1966, 1967; Encyclopaedia Eritannica Co., 1968, 1969, 1971.

2. M. E. Senko, "Infcrmation Storage and Retrieval," Advances in Information Systems Science, 2, (Ed. by J. T. Tou), Plenum Press, 1969, pp 229-281.

3. W. A. Woods, "Procedual Semantics for Question Answering", Proc AFIPS, 33 (FJCC 1968), pp 457-471.

4. C. A. Montgomery, "Is Natural Language an Unnatural Query Language?" Proc ACM, 25, (August, 1972), pp 1075-1078.

5. N. Chomsky, Syntactic Structures, Mouton & Co. 1957.

6. C. H. Kellog, "A Natural Language Compiler for On-line Data Management," Proc AFIPS, 33 (FJCC, 1968), pp 473-492.

7. C. S. Smith, "Ambiguous Sentences with AND", in Modern Studies in English, Ed by D. A. Reibel and S. A. Schane, Prentice Hall, 1969, pp 75-79.

8. L. R. Gleitman, "Coordinating Conjunctions in English", Language, 41, 1965 (Also in Modern Studies in English, see Reference 7, pp 80-112).

9. G. Salton and M. E. Lesk, "The SMART Automatic Document Retrieval System -- an Illustration", Comm ACM, 8,6 (June 1965), pp 391-398.

10. T. Winograd, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", Project MAC, MAC TR-84, February, 1971.

11. B. H. Dostert, "REL - An Information System for a Dynamic Environment", REL Report 3, California Institute of Technology, December, 1971.

12. B. W. Puerling and J. T. Roberto, "The Natural Dialogue System", to appear in BSTJ.

13. N. R. Sinowitz, "DATAPLUS -- A Language for Real Time Information Retrieval from Hierarchical Data Bases", Proc AFIPS, 32 (SJCC 1968), pp 395-401.

14. J. M. Wier, "Interactive Information Management Systems", Bell Sys. Tech. J., 52,10 (Dec. 1973), pp 1681-1690.

15. L. E. Heindel and J. T. Roberto, "The Off-the-Shelf System -- A Packaged Information Management System", Bell Sys. Tech. J., 52,10 (Dec. 1973), pp 1743-1763.

16. T. A. Gibson and P. F. Stockhausen, "Master Links -- A Hierarchical Data System," Bell Sys. Tech. J., 52,10 (Dec. 1973), pp 1691-1724.

17. D. T. Chai, "An Information Retrieval System Using Keyword Dialog", Information Storage and Retrieval, 9,7 (July 1973), pp 373-387.