W. J. Plath

REQUEST: A Natural Language Question-Answering System

Abstract: REQUEST is an experimental Restricted English QUESTion-answering system that can analyze and answer a variety of English questions, spanning a significant range of syntactic complexity, with respect to a small *Fortune*-500-type data base. The long-range objective of this work is to explore the possibility of providing nonprogrammers with a convenient and powerful means of accessing information in formatted data bases without having to learn a formal query language. To address the somewhat conflicting requirements of understandability for the machine and maximum naturalness for the user, REQUEST uses a language processing approach featuring: 1) the use of restricted English; 2) a two-phase, compiler-like organization; and 3) linguistic analysis based on a transformational grammar. The present paper explores the motivation for this approach in some detail and also describes the organization, operation, and current status of the system.

Introduction

REQUEST [1, 2, 3] is an experimental Restricted English QUESTion-answering system that has been implemented in LISP 1.5 and runs under an interactive operating system in one million bytes of virtual storage. It is currently capable of analyzing and answering a variety of English questions, spanning a significant range of syntactic complexity, with respect to a small *Fortune*-500-type data base.

The general objective of this work is to investigate the feasibility of developing machine-understandable subsets of natural languages such as English which can serve as a basis for effective man-computer communication. More specifically, we are seeking to explore the possibility that such natural language subsets can provide nonprogrammers with a convenient and powerful means of accessing information in formatted data bases without having to learn a formal query language. In turning to natural language for this purpose, our central goal is to achieve maximum naturalness and flexibility for the user, who ideally should not have to learn any new linguistic conventions in order to interact successfully with the system. Accordingly, such familiar devices as lists of required words, lists of restricted words, and the use of fixed syntactic frames are explicitly avoided in REQUEST, because they all involve arbitrary conventions that must be consciously learned in much the same way that those of a formal language are. (In fact, we would claim that systems which make substantial use of such devices are not based on natural language to any significant degree, but instead, like COBOL, involve little more than a formal language thinly disguised by a layer of natural language vocabulary.)

In order to address the somewhat conflicting requirements of understandability for the machine on the one hand and maximum naturalness for the user on the other, the REQUEST system employs a language processing approach with three salient characteristics:

1. The use of restricted English;
2. Linguistic analysis based on a transformational grammar; and
3. A two-phase, compiler-like organization.

In the next section, the general nature of each of these fundamental features is briefly discussed. This is followed by a more detailed presentation of the reasons for employing a transformational grammar, which is the characteristic that most sharply distinguishes REQUEST from other natural language query systems. The remainder of the paper is devoted to an overview of REQUEST system organization and operation, a summary of current system status, and brief consideration of anticipated lines of future development.

Basic design features

The first basic design feature, the use of restricted natural English, is dictated by the realities of the present state of the art of formal description of natural languages: specifically, the fact that nothing remotely approaching a complete grammar or semantics of all of English (or of any other natural language) either exists now or appears likely to materialize in the near future. The REQUEST approach to restricted English (which is similar in certain respects to those adopted in the systems of Woods [4] and Winograd [5]) involves sharply

limiting the semantic scope of the English material to be covered. This is accomplished by focussing on one relatively well defined universe of discourse at a time, for example, the "world" of a business statistics data base. Having thus greatly restricted what the user can "converse" with the computer about, we then seek to provide him within that domain with a flexibility of syntactic and lexical expression approaching that of normal English.

Restricting a natural language subset in the manner just described has two major advantages: First, it reduces the semantic universe that must be handled to a size that is potentially tractable for purposes of formal analysis and "understanding" by a computer. Second, it leads directly to major nonarbitrary reductions in the range of vocabulary (and, to a lesser extent, the range of syntactic constructions) that must be covered in the subset, since there is no need to include words, constructions, or meanings of words not related to the subject matter the user will necessarily be dealing with.

In a research project such as that on REQUEST, one conceivable drawback of a narrow semantic focus is the possibility that solutions worked out for a specific domain of discourse may not be readily extendable to others. In the hope of minimizing such difficulties, we have chosen to work initially with the world of business statistics, because it appears to be representative of a large and important family of data bases involving periodic, numerical data. Despite the existence of idiosyncratic differences, the various members of this family—such as weather data, census data, and price and wage statistics—share a broad range of semantic relationships, including notions of time, comparison, and various higher order functions of the primitive data (e.g., sums, averages, ratios, rates, maxima, and minima). Because these shared semantic relationships tend to be expressed linguistically in a very similar manner for all of the domains in question, the prospects appear quite favorable for a substantial carryover of results from one case to the next.

The second major feature of REQUEST's approach to natural language processing is the treatment of input queries in restricted English as high-level-language expressions that are to be compiled into executable code. As in the case of compilers for formal languages, the process consists of two consecutive phases: a *parsing phase*, in which the structure of the input language expression is determined, and a *translation* (or semantic interpretation) *phase*, in which the resulting structural description is mapped into object language code. In REQUEST, the mechanics of the latter process closely resemble those employed in conventional compilers, in that they are based on a scheme originally proposed by Knuth [6] as a generalization of standard syntax-directed translation techniques. A similar degree of correspondence does not exist for the parsing phase, however, because we have designed it around transformational grammar, a form of linguistic description that differs markedly from anything used in compilers for formal languages.

The third basic design feature of REQUEST, employment of linguistic analysis based on a transformational grammar, was adopted in an attempt to deal with the complexity and diversity that are characteristic of even restricted subsets of natural language as we have defined them. The key properties of a transformational description are 1) the definition of two distinct levels of linguistic structure—*surface structure* and *underlying structure*—and 2) the specification of a formal mapping relating them. The nature of such grammatical models and their relevance to the problems of developing user-oriented subsets of natural language are now examined in some detail.

Motivation for using a transformational grammar

• *Inadequacy of surface structure models*

Within the field of linguistics, the principal impetus for adopting a transformational model for the grammatical description of natural languages has been recognition that models based exclusively on surface structure are inadequate both 1) as a basis for defining grammaticality (i.e., membership vs nonmembership in the set of well-formed sentences) and 2) as a vehicle for representing systematic relationships that exist among natural language sentences, including those involving meaning equivalence. However, these objections do not carry over to the realm of formal languages, many of which can be adequately described in (context-free) surface structure terms using such devices as the familiar BNF notation.

A surface (phrase) structure description of a natural or formal language expression may be characterized informally as a representation of the hierarchical grouping of the elements of the expression into higher-order structures. As shown in Fig. 1, the representation typically takes the form of a labeled bracketing or tree that preserves the left-to-right ordering of the elements, which occupy the leaves of the tree. With the exception of the parentheses in the upper tree, each such terminal element is directly and exclusively dominated by a node whose label specifies the symbol category or part of speech to which the element belongs. Thus "A" is an expression (EXP), "×" is an operator (OP), "Does" in Fig. 1(b) is an auxiliary (AUX), and so on. The branching pattern of each tree reflects the way in which sets of categories combine to form subexpressions or phrases, which in turn combine into structures of successively higher scope, ultimately resulting in an expres-

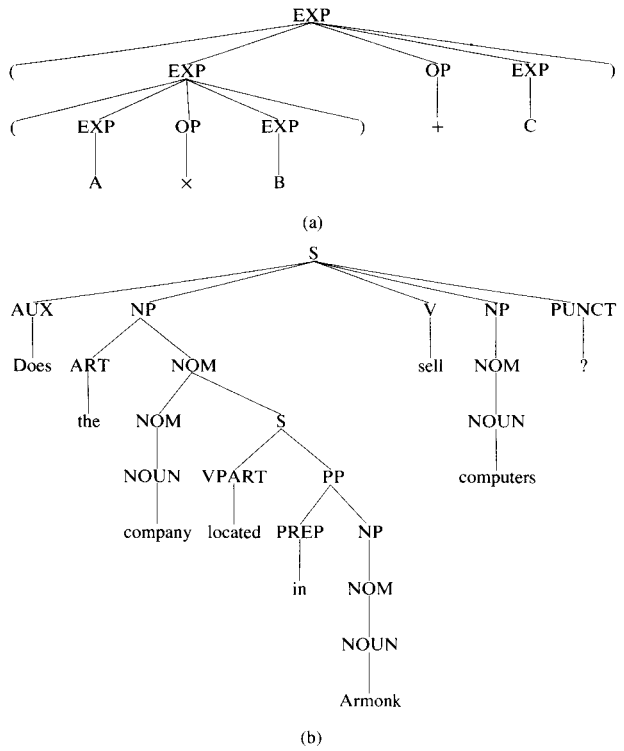

Figure 1 Surface structure (a) of "((A × B) + C)" and (b) of "Does the company located in Armonk sell computers?".

sion of the type indicated by the label of the root node – in Fig. 1(a) and (b), an expression (EXP) and a sentence (S), respectively.

In formal languages there is generally a close correspondence between surface structure and meaning. Accordingly, once the surface structure of an input expression has been determined during the parsing phase, the process of translating it correctly into object language code tends to be a relatively straightforward one. For natural languages, however, major divergences exist between the surface structure of sentences and the meanings those sentences convey, with predictable implications for the difficulty of surface-structure-based approaches to translation.

Two principal sources of divergence are the presence of "understood" elements in sentences and the existence of what may be called "structural synonymy" among sentences. Examples of the first phenomenon are given below in examples 1) – 3), where the (a) version of each sentence is perfectly natural and understandable to a speaker of English yet omits key meaning-bearing elements that are present in the corresponding (b) version.

1. (a) Where is IBM's headquarters?
   (b) Where is IBM's headquarters *located*?
2. (a) What were IBM's 1973 earnings?
   (b) What were IBM's 1973 earnings *equal to*?
3. (a) What companies employed more people in 1972 than IBM did?
   (b) What companies employed more people in 1972 than IBM *employed people in 1972*?

Because these understood elements are not present in the (a) versions, by definition they are not present in the surface structure and hence are unavailable to any surface-structure-based semantic interpretation procedure. (Proposals to circumvent this difficulty by requiring users to avoid sentences with understood elements are basically unworkable because this would not only mean making the user give up the more compact (and generally more natural) form of expression but, in the case of examples like 3), would force him to go against habitual patterns of language behavior whereby understood elements are obligatorily deleted in certain constructions.)

Structural synonymy, the second major source of divergence, involves systematic meaning equivalences attributable not to the substitution of one synonymous word for another (i.e., lexical synonymy), but of one synonymous construction for another. An example of the syntactic diversity that can be associated with the latter phenomenon in English is given in example 4), which displays a single set of structurally synonymous expressions.

4. (a) IBM's earnings in 1972
   (b) the earnings of IBM in 1972
   (c) IBM's 1972 earnings
   (d) the 1972 earnings of IBM

(e) the amount $\left\{\begin{matrix}\text{of money}\\ \varnothing\end{matrix}\right\}\left\{\begin{matrix}\text{which}\\ \text{that}\\ \varnothing\end{matrix}\right\}$ IBM earned in 1972

(f) the amount $\left\{\begin{matrix}\text{of money}\\ \varnothing\end{matrix}\right\}\left\{\left\{\begin{matrix}\text{which}\\ \text{that}\end{matrix}\right\}\text{was}\right\}$ earned by IBM in 1972

Expressions 4(a) and 4(c) differ from 4(b) and 4(d), respectively, by the use of the preposed genitive "IBM's" instead of the postposed prepositional phrase "of IBM" to denote possession. Similarly, expressions 4(c) and 4(d) differ from 4(a) and 4(b) through use of the time compound construction "1972 earnings" instead of an expression containing the postposed prepositional phrase "in 1972." In the last two cases in 4), the nominalized form "earnings," which appears in 4(a) – 4(d), has been replaced by a corresponding relative clause construction involving the verb "*earned*"; as a transitive verb, the latter may occur in either the active 4(e) or the passive voice 4(f).

Taking into account the various options indicated by the braces, 4) can be seen to contain no less than six-

teen syntactically different ways of expressing the same semantic content—an impressive number, but one by no means atypical of English. Each variant has a distinct surface structure resulting from a unique combination of word order, choice of function words, choice of grammatical endings, and the presence or absence of potential understood elements. These formal differences pose severe problems for any surface-structure-based semantic interpretation procedure that attempts to deal with all of the variants or even with a representative selection thereof. Moreover, whereas there undoubtedly are differences in the relative frequencies of occurrence of the sixteen alternatives, none of them is a priori so clearly unusual as to be an obvious candidate for elimination from a natural language subset. Thus there appears to be no simple way of rescuing surface-structure-based semantic interpretation without simultaneously abandoning the goal of flexibility and naturalness for the user.

• *Underlying structure*

To surmount difficulties of the sort just described, the REQUEST system uses a transformational grammar to provide linguistic descriptions at the underlying structure level as well as at the level of surface structure. The underlying structures assigned by the grammar are considerably more abstract than their surface structure counterparts and come much closer to providing an adequate representation of those elements of meaning that are relevant for semantic interpretation. Some of the properties of this deeper level of representation are illustrated in Fig. 2, which displays the underlying structure assigned to the question "Is IBM's headquarters in Armonk?" by the current grammar.

As can be observed from the figure, our underlying structures are trees consisting of one or more nested propositions (S1), each marked off by a pair of boundary symbols (BD). Each proposition consists of a predicate (V) followed by its associated arguments (NP's), which always occur in a fixed order. Such surface structure elements as auxiliaries, prepositions, inflectional endings, and punctuation—all major sources of syntactic variation—have been eliminated in favor of binary syntactic features (e.g., (+ PAST) and (− PAST) on the top S1 node to represent past and present tense, respectively; (+ QUES) on the top S1 node to mark interrogative sentences; and (+ SG) and (− SG) on NOUN nodes to indicate singular and plural number, respectively). At the same time, understood elements, such as the main predicate *located*, are systematically restored. (The V nodes dominating underlying predicates carry feature information relating both to properties of the predicates themselves (e.g., whether or not they can be realized as adjectival elements (+ ADJ) in surface structure) and to properties of their NP arguments—e.g., the

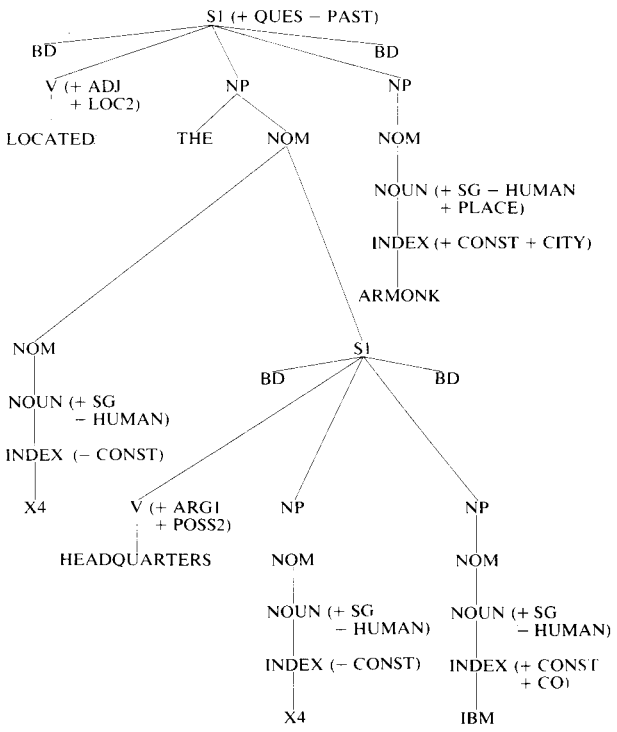

Figure 2 Underlying structure of "Is IBM's headquarters in Armonk?".

requirement that the second argument be a locative (+ LOC2) or a possessive (+ POSS2) noun phrase.)

The combined effect of elimination of syntactic variation and of restoration of "understood" elements is to provide a sort of canonical form for sentences, where each underlying structure in general corresponds to a *set* of structurally synonymous surface variants. Thus, for example, the structure displayed in Fig. 2 is in fact assigned to all of the variants listed in 5), thereby providing a common basis for their processing during the translation phase.

5. (a) Is IBM's headquarters in Armonk?
   (b) Is IBM's headquarters located in Armonk?
   (c) Is the headquarters of IBM in Armonk?
   (d) Is the headquarters of IBM located in Armonk?

Stripped down to its bare essentials, the tree of Fig. 2 can be reduced to the bracketed expression 6), revealing another key property of our underlying structures—their close resemblance to expressions in the predicate calculus. As illustrated in Fig. 2, this parallelism includes the representation of surface proper nouns (e.g., "IBM" and "Armonk") as logical constants (INDEX (+ CONST)) and the treatment of surface common nouns such as "headquarters" as variables (INDEX (− CONST)) in propositional functions. These same variables and constants play a central role in the executable code that is the output of the REQUEST system's translation phase

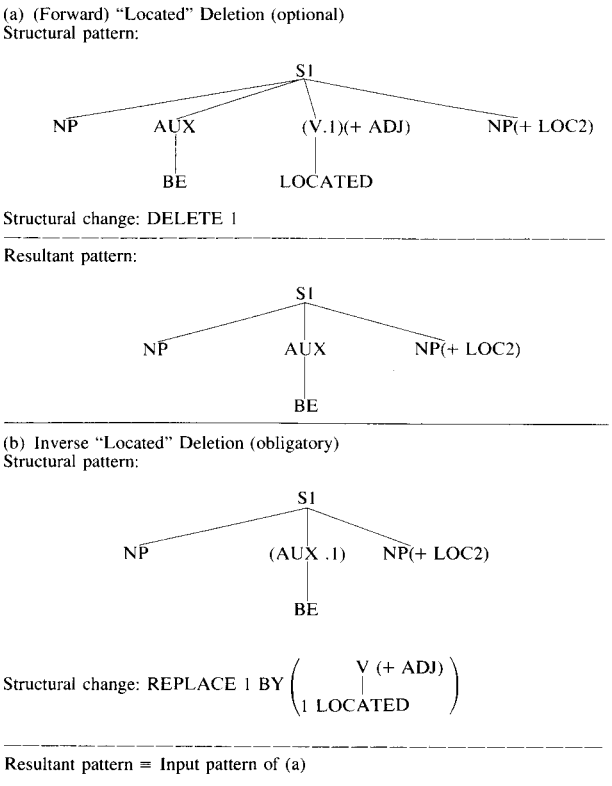

Figure 3 A transformation and its inverse.

(the so-called "logical form" of the sentence); accordingly, their presence in underlying structures (which serve as input to that phase) contributes significantly to simplicity of processing at that point.

6. [LOCATED (THE X4 [HEADQUARTERS (X4) (IBM)]) (ARMONK)]

• *Transformational grammar*

As noted previously, a transformational grammar formally specifies the relationship between two levels of linguistic structure: the underlying level and the surface level. The correspondence between the levels is described in terms of an ordered set of tree-mapping rules, or *transformations*, each of which defines an incremental structural change required in the process of passing from one level to the other. Because most linguists are not concerned with applications involving sentence parsing, they have tended to describe such transformational processes as proceeding *generatively* from underlying structure to surface structure, in a manner somewhat analogous to that in which theorems are derived from a set of underlying axioms in a formal deductive system. REQUEST's transformational parser necessarily proceeds in the opposite direction, however, first computing the surface structure of an input query and then transforming it step by step into a corresponding underlying structure. Consequently, the REQUEST transformational grammar uses rules that are the inverses of conventionally oriented grammatical transformations. (The parser also contains the option of employing "forward" rules in checking the validity of inverse derivational paths, a useful feature when debugging revisions to the grammar.)

The statement of each rule in the REQUEST grammar, whether it is a forward rule (i.e., the sort conventionally defined by a linguist) or a corresponding inverse, is expressed in a pattern-action format, as illustrated in Fig. 3. Forward "Located" Deletion (Fig. 3(a)) is a transformation that *optionally* deletes the adjectival predicate "located" from a certain class of English sentences (actually, from the trees representing those sentences at a point intermediate between the surface and underlying structure levels), thereby accounting for the occurrence of such structurally synonymous pairs as 5(a) – 5(b) and 5(c) – 5(d). The *structural pattern* of the rule specifies its domain of applicability: namely, any clause (S1) consisting of an arbitrary subject noun phrase (NP), a form of the auxiliary BE, the adjectival predicate "located," and an arbitrary locative noun phrase (NP (+ LOC2)) in that order. The *structural change* specifies the action that is to be carried out on a tree if the pattern is matched and the option of applying the rule is taken – in this case, a simple deletion of the subtree corresponding to the pattern element labeled "1."

The portion of Fig. 3(a) labeled "resultant pattern" represents the general tree configuration resulting from application of the forward rule. Although not properly part of the rules that produce them, such resultant patterns are important, because they serve to define the domains of corresponding inverse transformations. Thus, the structural pattern of Inverse "Located" Deletion (Figure 3(b)) can be seen to be identical (except for the addition of a numerical label) to the resultant pattern of the forward rule. The structural change of the inverse is then defined in such a way as to undo the effect of the forward transformation: in this case, by specifying replacement of the auxiliary (i.e., the subtree labeled "1") by itself and an instance of the missing predicate "located."

REQUEST system organization

The REQUEST system consists of a set of LISP programs and an associated set of data files containing the lexicon, grammar, semantic interpretation rules, and data base. As shown in Fig. 4, these elements are organized into a system with two major components – one transformational, the other interpretive – broadly corresponding to the parser/translator organization of a compiler.

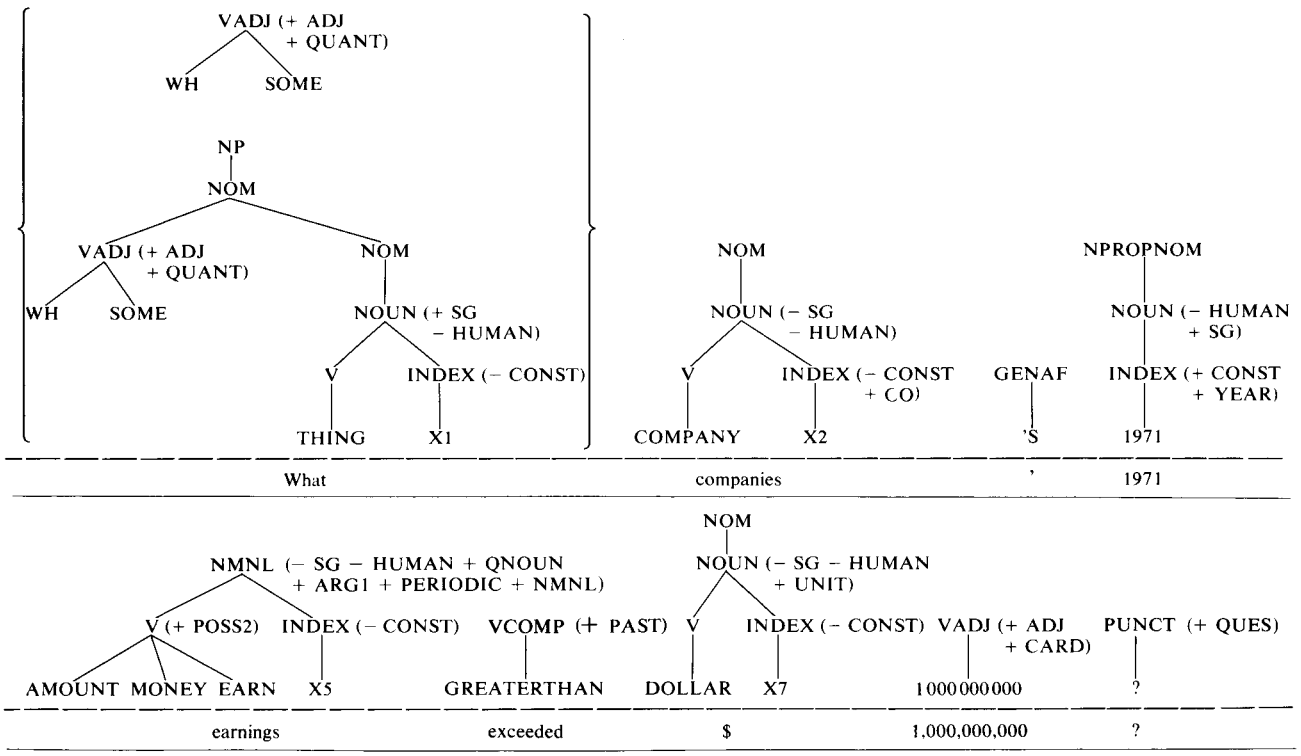

Figure 4 Overall system organization.

The transformational component analyzes input word strings and computes their underlying structures; it consists of two main parts: a preprocessor and a parser. The interpretive component also has two main subcomponents: 1) a Knuth-style semantic interpreter [2, 7, 8], which translates each underlying structure into a corresponding *logical form*, i.e., a formal expression specifying the configuration of executable functions required to access the data base and compute the answer to the original question; and 2) a retrieval component, which contains various data-accessing, testing, and output formatting functions needed to evaluate the logical form and complete the question-answering process. (Implementation of the interpreter is due to S. R. Petrick who has also devised the specific semantic interpretation rules employed in REQUEST. F. J. Damerau is responsible for the design and implementation of the current retrieval component.)

• *Transformational component*

Within the transformational component, the preprocessor partitions the input string into words and punctuation marks and then looks up each segment in the lexicon, producing a *preprocessed string* of lexical trees, which serve as input to the parser. Multi-word strings that function as lexical units are identified by a "longest match" comparison with a special phrase lexicon; whereas the lexical trees for arabic numerals (which may variously represent cardinals, ordinals, or year names) are supplied algorithmically rather than by lexical lookup. Whenever there is a gap in the preprocessed string, due to the presence in the input of misspellings, unknown words, ambiguous pronoun references, and the like, the preprocessor prompts the user to supply the required information.

The nature of the output of the preprocessor is illustrated in Fig. 5, which displays the lexical trees produced for the question "What companies' 1971 earnings exceeded $1,000,000,000?" in parallel with the corresponding elements of the original input string. (The lexical trees are represented internally in an equivalent parenthesized form, which is also used for the tree patterns in transformational rules.) As can be observed from the figure, the preprocessing phase of REQUEST does considerably more than assigning words to conventional part-of-speech categories, because it must set the stage as far as possible at the lexical level for the eventual mapping into underlying structures. Thus proper nouns such as "1971" and "IBM" are already treated as constants (INDEX (+ CONST)); whereas common nouns such as "earnings" appear as combinations of variables (INDEX (− CONST)) and underlying predicates. (The specific values of index variables, such as the "X1," "X2," "X5," and "X7" in Fig. 5, are used to keep track of

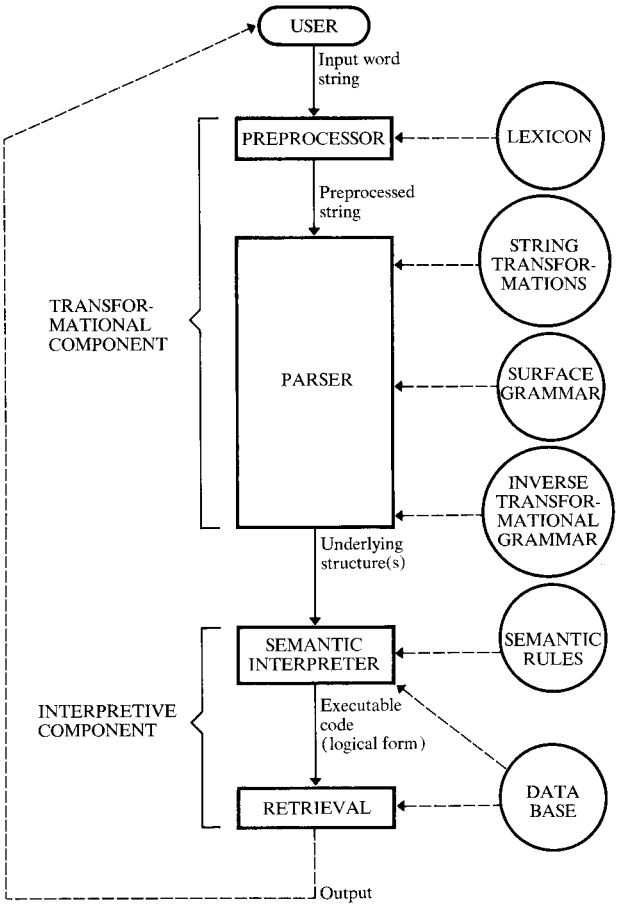

Figure 5 Output of the preprocessor in tree representation.

matters of reference in more complicated sentences by having the preprocessor assign identical variables to relative pronouns and their antecedents. No such complexity occurs here, however, and the preprocessor simply employs the word number of each common noun in manufacturing a unique variable name.) A further important effect of preprocessing, not evident from a single example, is the mapping of *lexical* synonyms onto a single lexical tree; e.g., any of the synonyms "net," "net earnings," "profit(s)," "net profit(s)," and "net income" would be assigned the same lexical tree as "earnings."

The transformational parser, whose original design and implementation are due to Petrick [9], is extremely general and could be employed not just for other subsets of English, but for subsets of any language described in transformational terms. (The version currently in use in REQUEST is the result of significant revisions and extensions by M. Pivovonsky, who has also been chiefly responsible for implementing the preprocessor.) Operation of the parser comprises three successive stages: 1) application of string transformations, 2) surface structure parsing, and 3) mapping of surface structures into underlying structures. In the first stage, the preprocessed string is successively analyzed with respect to the structural patterns of each of a linearly ordered list of *string transformations*. These rules share the property of being definable on local segments of the preprocessed string without reference to higher-order structures such as clause boundaries. Their principal functions include resolution of homographs, simplification of idiomatic structures, and the prevention of various artificial ambiguities at the surface structure level. (For a more detailed discussion of the nature and use of string transformations, see [3].) Each successful match against a string transformation leads to modification of one or more of the trees in the preprocessed string through application of the operations specified in the structural change of the rule in question. (These operations are drawn from precisely the same inventory of elementary transformations that REQUEST makes available for the processing of full trees by conventional forward and inverse transformations: deletion, replacement of a tree by a list of trees, Chomsky adjunction, feature insertion, and feature deletion.)

Upon completion of the string transformation phase, the resulting transformed preprocessed string—still in the form of a list of trees—is passed on to a context-free parser that computes the surface structure(s) of the input query. (Although one major effect of the use of string transformations has been a significant reduction in the number of unwanted surface parses, cases still occur where more than one surface parse is produced.) For the example in Fig. 5, the transformed preprocessed string, whose essential structure is conveyed by the bracketed set of terminal elements 7), is assigned a unique surface structure with the bracketed terminal string 8).

7. ((WH SOME) (COMPANY X2) 'S (1971 ((AMOUNT MONEY EARN) X5)) GREATERTHAN 1000000000 (DOLLAR X7) ?)

8. ((((WH SOME) (COMPANY X2)) 'S) (1971 ((AMOUNT MONEY EARN) X5))) GREATERTHAN (1000000000 (DOLLAR X7)) ?)

In the third and final stage, the transformational parser takes each surface structure in turn and attempts to map it step by step into the corresponding underlying structure according to the rules of the inverse transformational grammar. In this process, transformational inverses are applied in an order precisely opposite to that in which their forward counterparts would be invoked in sentence generation. (That is, inverses of postcyclic transformations are applied first, starting with the latest

and ending with the earliest; then inverses of the cyclic transformations are applied (also in last-to-first order) working down the tree from the main clause, until the most deeply embedded clauses have been processed.) For our running example this results in the creation of an underlying structure tree with the bracketed terminal string 9).

```
9. (BD GREATERTHAN (THE (X5
   (BD AMOUNT X5 ((MONEY X39)
   (BD EARN ((WH SOME) (COMPANY X2))
   X39 1971 BD)) BD))) 1000000000
   (DOLLAR X7)) BD)
```

• *Interpretive component*

At the completion of the transformational parsing phase, the resulting underlying structure is passed to the Knuth-style semantic interpreter. As described in greater detail in [2], the interpreter maps the underlying structure into a corresponding logical form through the systematic application of a series of semantic interpretation rules. These rules are in the form of sets of translation equations, one for each of the local branching patterns (node plus immediate descendants) that are permitted to occur in underlying structure trees. Each such pattern is represented as a list consisting of the name of the parent node followed by the names of its immediate descendants listed in left-to-right order as they occur in the tree. Example 10a) illustrates this convention as applied to the branching pattern for relative clauses, which in our grammatical description always consists of a NOM node immediately dominating another NOM node followed by an S1 node.

```
10. (a) (NOM NOM S1)
    (b) (((V 0) (UNION (V 1) (V 2)))
        ((N 0) (ADDRESTR (N 1) (N 2)))
        ((SG 0) (SG 1)))
```

The associated translation equations 10b) describe the way in which values of specified attributes of nodes in the configuration are to be assigned. The numerals in each translation equation designate nodes in the corresponding branching pattern, which are treated as though they were numbered consecutively from the left, starting with zero. The codes that are paired with the numerals denote the values of specific attributes; thus (V 0) denotes the value of the attribute V of node 0 (the parent NOM node), (N 2) denotes the value of the attribute N of node 2 (the S1 node), etc. In each translation equation, the nodal attribute designated at the left is defined as the function of nodal attributes specified by the expression on the right. Thus in 10b), the V attribute (i.e., the list of free variables) of node 0 is defined as the union of the V's of nodes 1 and 2; the N attribute (i.e., the partially constructed logical form) of node 0 is defined as a special function ADDRESTR of the N's of its immediate descendants; and the SG attribute (grammatical number) of node 0 is defined as identical to that of node 1. Although the three translation equations of 10b) all have the effect of passing information up the tree, this is by no means universally the case: There are also rules that pass information down the tree, and information can be passed laterally among sibling nodes as well.

As in the case of the transformational parser, the computational algorithms used to apply the semantic interpretation rules are entirely application independent. The same does not hold true for the rules themselves, however. Whereas certain important rules, such as those for processing relative clauses 10b) and questioned noun phrases, can be expected to remain unchanged from one application to the next, others (notably those low-level rules relating to the translation of individual underlying predicates, such as "EARN") are highly application specific.

The nature of the output of the semantic interpreter in our current application is illustrated in 11), which displays the logical form produced for our running example "What companies' 1971 earnings exceeded $1000000000?".

```
11. (setx 'X2
      '(and
        (forall 'X50
          (setx 'X39
            '(testfct X39 '('NET_INCOME
                                X2 '1971) NIL))
          '(greaterthan X50 '1000000000))
        (company X2)))
```

The output is in the form of a LISP S-expression containing an array of functions that are evaluated during the retrieval phase in order to answer the question. The top-level *setx* function indicates that the desired answer is in the form of a set of objects X2 each member of which satisfies the specification that follows. In this case, the latter is in the form of a conjunction of two terms: the universally quantified expression beginning "(*forall* 'X50") and the expression "(*company* X2)," which stipulates that the X2's under consideration are companies. The quantified expression indicates that all members X50 of the set of 1971 earnings figures X39 of the companies in question must exceed 1000000000.

Some of the functions that appear in logical forms—e.g., *and* and *greaterthan*—are already supplied by LISP; while others, such as *setx*, *forall*, and *testfct*, have had to be defined in LISP in order to provide the required mechanisms for data base accessing, testing of values, and output formatting during the retrieval phase. In the case

of 11), interpretive execution of the logical form during that phase produces the final output 12), which is displayed at the user's terminal.

12. ANSWERS:
    1: GENERAL MOTORS
    2: EXXON
    3: IBM

Current status

The REQUEST system has been in experimental operation for nearly three years, a period during which a series of extensions of linguistic coverage and improvements in system capabilities and organization have been made. The current grammar contains more than one hundred transformational rules, which provide for a wide variety of basic English constructions—including wh- and yes-no questions, relative clauses, genitives, negatives, locatives, and time expressions—as well as for selected phenomena more narrowly oriented towards the family of data bases under consideration (e.g., numerical quantifiers and rank expressions). A list of example questions partially illustrating the scope of current coverage is given in the Appendix.

Although the subset of English presently handled offers considerable flexibility of expression within the limited universe of discourse addressed, it is still quite inadequate with respect to coverage of certain complex phenomena—including comparison, conjunction, and quantification—which appear to be of central importance in providing users with a semantically powerful subset of English. Accordingly, such constructions have been the focus of our grammar development effort for several months and can be expected to remain so for some time in the future. This activity has already resulted in substantial extensions in the coverage of comparatives and in the addition to the parser of new pattern-matching and tree-mapping primitives.

The current REQUEST data base has also grown substantially within the past two years, but it is still relatively small: some twelve fields of Fortune-500-type information for about sixty major companies over a six-year period (1967–73). During the same period, both the retrieval component and the logical forms that serve to drive it have been extensively revised in two key respects—partially illustrated in 11) above—: the introduction of quantified expressions and the replacement of numerous special purpose functions by a few very general test functions. These changes have resulted in an interpretive facility that is both cleaner and more powerful than its predecessors. Moreover, in anticipation of planned future experiments with a different data base, further measures have recently been undertaken to separate data-base-dependent code from non-data-base-dependent code in the retrieval component, so that it will be as insensitive to such changes as are the parser, the preprocessor, and the semantic interpreter.

Future directions

As currently conceived, future development of the REQUEST system is expected to center about three closely related types of activity: 1) extension of grammatical coverage, 2) experimentation with one or more new data bases, and 3) exploration of the effectiveness of restricted English subsets as interaction languages through the medium of tests on actual users. As indicated in the previous section, efforts are already underway to provide future users with greater semantic power through a more comprehensive treatment of constructions involving comparison, conjunction, and quantification. Other major linguistic phenomena that we hope eventually to cover are ones relating to arithmetic predicates (sum, average, ratio, rate, etc.), connected discourse (including problems of pronominal reference), and the provision of definitional capabilities.

Our plans to work with a second data base are principally motivated by two considerations. One is the need to make a first-hand assessment of just what such a changeover entails with respect to revision and extension of various subparts of REQUEST, including the lexicon, grammar, semantic interpretation rules, and data accessing and formatting functions. The other is our desire to deal with a data collection whose contents are likely to be of active and continuing interest to some community of actual users—sufficiently so that they will be willing to participate on an experimental basis in the development and testing of a version of the system with the appropriate semantic orientation. Based on our investigations to date, the most promising candidates currently available for this purpose appear to be machine-readable files on land use and related topics (assessments, property sales, school census, etc.); these files contain material of great interest both to such officials as city planners and assessors and to members of the taxpaying public. A joint study agreement providing a framework for working with such a data base and its associated user community has recently been concluded with a nearby municipality, and initial steps towards extending REQUEST to cover the domain of land use are now under way.

References

1. W. J. Plath, "Transformational Grammar and Transformational Parsing in the REQUEST System," *Research Report 4396*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1973. (to appear in A. Zampolli (ed.), *Computational and Mathematical Linguistics. Proceedings of the International Conference on Computational Linguistics*, Pisa 27 VIII–1/IX 1973, Casa Editrice Olschki, Firenze, Vol. I.)

2. S. R. Petrick, "Semantic Interpretation in the REQUEST System," *Research Report 4457*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1973. (to appear in A. Zampolli (ed.), *Computational and Mathematical Linguistics. Proceedings of the International Conference on Computational Linguistics*, Pisa 27 VIII–1/IX 1973, Casa Editrice Olschki, Firenze, Vol. II.).
3. W.J. Plath, "String Transformations in the REQUEST System," *American Journal of Computational Linguistics*, Microfiche 8 (1974). (Also appeared as *Research Report 4947*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1974.)
4. W. A. Woods, R. M. Kaplan, and B. Nash-Webber, "The Lunar Sciences Natural Language Information System," *Final Report BBN 2378*, Bolt Beranek and Newman Inc., Cambridge, Mass., 1972.
5. T. Winograd, "Understanding Natural Language," *Cognitive Psychology* **3**, 1 (January 1972).
6. D. E. Knuth, "Semantics of Context-free Languages," *Mathematical Systems Theory* **2**, 127 (1968).
7. S. R. Petrick, "On the Use of Syntax-based Translators for Symbolic and Algebraic Manipulation," *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, edited by S. R. Petrick, Association for Computing Machinery, New York, 1971, p. 224.
8. S. R. Petrick, "Mapping of Linguistic Structures into Computer-Interpretable Form," *AFCRL-TR-0055 Final Report*, Contract No. F19628-72-C-0129, Air Force Cambridge Research Laboratories, Bedford, Mass., 1972.
9. S. R. Petrick, "Transformational Analysis," *Natural Language Processing*, edited by R. Rustin, Algorithmics Press, New York, 1973, p. 43.

Appendix: Some examples of current linguistic coverage

1. Is IBM's headquarters (located) in Armonk?
2. What city is the headquarters of IBM (located) in?
3. Are the headquarters of the IBM Corporation in (the city of) Chicago, Illinois?
4. Did Chrysler make a profit in 1969?
5. When wasn't Chrysler profitable?
6. How much (money) did GM gross in 1967?
7. What company ranked fifth in 1971 sales?
8. What was Xerox's rank with respect to growth rate in 1970?
9. What company was 16th in earnings in 1972?
10. How (high) did Exxon rank in 1973 sales?
11. How many companies' headquarters are (located) in New York?
12. In how many years was General Foods profitable?
13. What companies' 1973 sales were greater than GM's 1973 earnings?
14. How large a number of people did companies in Chicago employ in 1969?
15. What companies (that are) not (located) in New York City are (located) in New York State?
16. What companies whose sales exceeded $2,000,000,000 in 1970 were unprofitable in 1970?
17. How large were the 1972 sales of the companies whose growth rates in 1970 exceeded ten percent?
18. How large were the workforces of IBM, GE, and Xerox in the years 1970, 1971, 1972 and 1973?
19. How many people were employed by GM, Ford and Chrysler during the period from 1967 through 1972?
20. What were the top ten companies in sales in the year 1970?
21. How large were the 1972 earnings of the companies ranking 10th through 15th in 1969 sales?
22. Were IBM's 1973 earnings greater than Mobil's?
23. What companies' sales for 1970 were as large as Chrysler's for 1971?
24. Roughly how many workers were employed by GM in 1969?
25. What companies made at least $1,000,000,000 in 1973?
26. Exactly what amount (of money) did MMM earn in 1968?
27. What companies' 1969 sales were about the same size as IBM's?
28. IBM's 1968 sales?
29. The 1969 sales ranks of companies in Detroit?
30. List the companies (that are) (located) in Michigan!
31. Print out the 1967–1971 earnings of Ford and GM!

Received April 2, 1975; revised February 23, 1976

The author is located at the IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y. 10598.