A natural language compiler for on-line data management

by CHARLES H. KELLOGG
System Development Corporation
Santa Monica, California

INTRODUCTION
During the past few years there has been a rapid advance in the technology of time-sharing systems and software to permit quick access to large files of structured data. This has led to a growing interest in communicating with computer files directly in a natural language such as English. The natural language systems described in the literature are largely small-scale research vehicles dealing with small data bases of restricted subject scope. Giuliano (1965), among others, has questioned the generalization of these systems to wider universes of discourse. Developments in this area have been reviewed by Simmons (1966), and by Bobrow, Fraser and Quillian (1967). In contrast, the work in on-line data management has been more concerned with the efficient organization of structured data to allow for quick access and maintenance of large volumes of formatted information [see the reviews by Kellogg (1967), Climenson (1966), and Minker and Sable (1967)].

At SDC, we have been concerned for some time with the design and implementation of a prototype system that combines the advantages of the two approaches: the natural language system's facility for accepting ordinary English, and the on-line data management system's ability to manipulate large quantities of data of varying subject matter. We have been particularly concerned with striking a reasonable compromise between the difficulties of allowing completely free use of ordinary English and the restrictions inherent in existing artificial languages for data base description and querying.

This paper will discuss a number of the properties that seem necessary and desirable for an on-line English subset. We will describe the implementation of an English subset and the construction of a compiler that translates facts and requests for facts, expressed in ordinary English grammatical patterns, into procedures in an intermediate language (IL)—procedures that are sufficiently explicit to be directly accepted and processed by a data management machine. The use of computer-to-user feedback to ameliorate some of the confining restrictions of the English subset will be illustrated, as will the means whereby the user can extend the vocabulary and scope of the English subset to permit the description and interrogation of data bases of varying structure and content.

We will show how semantic structures can be precisely represented as procedures and how, from this point of view, a natural language compiler constitutes a device for transforming natural language syntactic structures into underlying semantic structures. Along the way we will draw upon a number of examples illustrating the on-line use of the most recent version of the prototype system.

Research methodology
The components of an experimental system (called CONVERSE) for investigating the feasibility of natural language compilers and for constructing and testing such compilers are shown in Figure 1. The programs and associated files are implemented on the IBM Q-32 Time-Sharing System at SDC.

The META-LISPX meta (or syntax-directed) compiler developed by Book and Schorre (1965) has been used to implement, test and modify the natural language compiler and data management
FIGURE 1—The CONVERSE system

machine. META-LISPX is implemented within a version of the LISP 1.5 List Processing System. It accepts a description of an object compiler expressed as a series of phrase structure rewrite rules. These META rules are associated with procedures, written in LISP, that represent the actions to be taken once a substring or constituent of an input sentence has been recognized. Since these actions can take the form of any procedure expressible in the LISP list processing language, they can be of any desired degree of generality or complexity. In particular these procedures implement a process of semantic interpretation—a process that is controlled or directed by the recognition of syntactically well formed constituents. The object natural language compiler produced by the META compilation process contains a top-to-bottom left-to-right syntactic recognition algorithm which is driven by the encoded form of the phrase structure rules input in the META-LISPX language. As illustrated in Figure 1, a data management machine has also been produced using META-LISPX. These object programs are easily modified or extended by incremental recompilation.

In use, the natural language compiler first constructs and then uses a file of general facts to control the compilation process. It also updates and utilizes information in the dictionary that it shares with the data management machine. Procedures produced by the compiler drive the machine to store, modify, or search a file of specific facts.

The difficulty in using a meta- or syntax-directed compiler to implement a subset of natural language has been recognized by Cheatham and Warsall (1962). They point out that, in the case of translating a programming language input string into machine language, the formation rules of the programming language have semantic significance. Consequently, given a parse of the input string the semantic interpretation of this string is relatively straightforward. They go on to discuss the possibility of phrasing formation rules for a subpart of English in a form acceptable to a syntax-directed compiler: "however in contrast to the case of algebraic compilers an application of one of these rules does not in general correspond to a semantically meaningful act." Their point is well taken. As we shall see the recognition algorithm built into a syntax-directed compiler must be supplemented by a rather considerable amount of semantic interpretation apparatus in order to deal with the complexities (such as ambiguity resolution) of even a small subset natural language.

There are six basic actions which we require from a natural language compiler:

1. An anomalous input sentence, one lying outside of the English subset, must cause the compiler to construct appropriate feedback messages and send this information to the user. This information serves two important purposes. First, it will make the user aware of the current limitations of the Eng-
lish subset, and secondly, it should enable him to take constructive steps to enrich the English subset. The feedback should give the user information as to how and why the English subset was exceeded.

2. Sentences which are syntactically or semantically ambiguous must be recognized as such and, where possible, the compiler should resolve this ambiguity. For example, semantic information can often be used to resolve cases of syntactic ambiguity and vice versa. When this is impossible the ambiguous interpretations must be displayed to the user in the intermediate language format.

3. A declarative sentence specifying general information must cause the natural language compiler to update its own store of such data and create appropriate dictionary entries for newly recognized words.

4. A declarative sentence specifying specific information must be translated into explicit intermediate language file storage procedures.

5. A declarative sentence specifying a definitional extension of the English subset vocabulary must lead to appropriate actions to extend the range of interpretations of existing words in the dictionary, or to add newly defined words to the dictionary.

6. The sixth basic action is to recognize interrogative sentences and to translate the requests implicit in these sentences into explicit file searching procedures.

Early experimental work on the CONVERSE system is reported in Kellogg (1967a, 1967b). At first (1967a) we developed an English subset only for questions and we used the QUUP artificial query language for the LUCID data management system as the intermediate language. Further, we translated questions into QUUP queries for only a single data base. Thus this initial experimental system was open to the same critical question asked of many experimental natural language systems, namely, "is it generalizable to other data bases of significantly different structure and/or content?"

Our next step was to allow some degree of generalization to data bases of different structure or content. This approach was based on a "computer interrogation of the user technique" (1967b). The user was led through a series of steps designed to obtain a description of his data base from the answers he provided to the computer displayed questions.

The body of this paper is concerned with the more general framework already alluded to, viz., one in which a user can input interrogative and declarative sentences in any order to query a data base, to modify the content of a data base, or to extend the range of semantic structures admissible with respect to that data base.

Properties of the source language and the target language

The function of the (English) source language is to provide an on-line user with the ability to communicate his data and requests in the ordinary grammatical patterns or syntactic structures of English. The function of the target (intermediate) language is to make explicit the semantic structure of the English sentence as a series of file storage and search procedures. Syntactic and semantic structures are characterized by syntactic and semantic categories and by the allowed associations or relations among these categories. These characterizations may be represented formally by the use of rewrite or formation rules.

Figure 2 illustrates some of the categories used in structuring the English subset and intermediate languages. Such traditional categories as noun phrase, verb phrase, adjectival phrase, prepositional phrase and the lower level categories: noun, adjective, preposition, verb and determiner are used to formally specify the grammar of our English subset. The underlying, or deep semantic structures that we recognize comprise such semantic categories as procedure, proposition, term, predicate, operator, function, relation, set, and property. Lower level semantic categories correspond to concepts such as: DEPART, SMOGGY, and STATE. The members of the lowest level syntactic categories are the words in the dictionary, while the members of the lowest level semantic categories are seen to be groupings of ordered N-tuples of semantic values in the data base.

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.
In particular the associations between English subset words and the semantic values in a data base may be many-to-many and complex. The task of a natural language compiler is to recursively decompose sentences into their structural components and then to somehow use this information to cause the recursive composition of semantic categories into higher level semantic structures. The difficulty is one of establishing the correct relationships between syntactic and semantic categories, and sorting these out in an appropriate way to produce semantic structures that represent the facts or requests implicit in an English sentence.

Our semantic categories largely correspond to just those basic elements used in modern mathematics and symbolic logic to state facts about the actual or some possible world. Namely, the notions of object, classes or sets of distinct kinds of objects, properties and relations that may range over objects, and functions, which when applied to objects evaluate to other objects. Experience in mathematics and logic has shown that such a basic set of elements are sufficient, along with a few other devices, for the formulation of scientific theories. Bohnert (1966) has gone farther and expressed the view that such elementary elements (as they are allowed for in the predicate calculus) constitute a reasonable framework for representing the "deep" structure of natural languages.

We are now in a position to draw a precise distinction between data base structure and data base content. Data base semantic content correlates with Carnap's (1956) notion of extensional meaning. Similarly, data base semantic structure corresponds to his notion of intensional meaning. For example, "smoggy" (the semantic category) designates a property of certain things. This is the intension of the concept "smoggy." On the other hand, the extension of "smoggy" comprises the list of objects in the data base that possess this property. The intensional or structural aspect of meaning will be represented in terms of general facts constituting a data base description. The data base content is the extensional aspect of meaning and it is represented as a series of specific facts. Thus a change in the data base structure is taken to mean a change in data base description or equivalently a change in the intensional meaning of the universe of discourse, while a

---

### Figure 2—Syntactic and semantic categories

From the collection of the Computer History Museum (www.computerhistory.org)
change in data base content constitutes a change in the extensional meaning of the universe of discourse.

A rough correspondence between some of the syntactic categories and semantic categories can be given as follows: In general, common nouns may have many-to-many associations with sets. For example, there may be a number of common nouns, each of which designates the same kind of objects in the data base. On the other hand, one common noun may designate several kinds of objects included in the data base. Again several proper nouns may specify the same object in the data base or one proper noun (such as "New York") may refer to objects in several different sets. Similarly, there may be a many-to-many association between the transitive verbs and some of the prepositions in the English subset, and the various relations specified for the data base. Many-to-many relations may also obtain between noun phrases and functions and between adjectives and properties.

The present grammar for the English subset consists of approximately 100 rules of the phrase structure type and a few syntactic transformation rules. The grammar admits a reasonably wide class of declarative and interrogative sentences but it has been kept relatively small by implementation of just those rules that allow us to recognize the syntactic clues essential for driving the semantic interpretation phase of the natural language compiler. The word classes include the determiners, adjectives, proper, common, and abstract nouns, adverbs, verbs, prepositions, question words, relative pronouns, and conjunctions. Several forms of relative and dependent clauses, and preposition, noun, verb, adjectival and adverbial phrases are recognized. The example given later in the paper will provide a reasonably good idea of the range of acceptable sentences.

A formal specification of the rules for the intermediate language is shown in Figure 3. In designing this language, we have been guided by our earlier experience in translating questions into queries admissible to the LUCID data management system and by three principal design objectives:

1. The language must be procedural. It should be comparable to procedure-oriented programming languages with regard to implementation and independence from particu-

<table>
<thead>
<tr>
<th>Rule</th>
<th>Formation Rules for the Intermediate Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIM</td>
<td>( \text{LIM} \rightarrow \text{LIM} )</td>
</tr>
<tr>
<td>V</td>
<td>( \text{V} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>ID</td>
<td>( \text{ID} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>D</td>
<td>( \text{D} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>VQ</td>
<td>( \text{VQ} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>V</td>
<td>( \text{V} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>N</td>
<td>( \text{N} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>P</td>
<td>( \text{P} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>R</td>
<td>( \text{R} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>E</td>
<td>( \text{E} \rightarrow \text{NP} )</td>
</tr>
<tr>
<td>PFP</td>
<td>( \text{PFP} \rightarrow \text{NP} )</td>
</tr>
</tbody>
</table>

Note: Terminal symbols appear in quotes. Brackets indicate a choice of elements. Parentheses indicate optional elements. An asterisk indicates an element may be repeated.

FIGURE 3—Formation rules for the intermediate language.
lar machine configurations or detailed information structure considerations.

2. It should be a powerful and expressive language for stating both storage and search procedures. In this regard, we have been influenced by the structure of the predicate calculus. The intermediate language permits the composition of functions, the nesting of relations, the embedding of procedures within procedures, and quantification over sets.

3. The language should be relatively easy to read and understand.

The basic components are:

- letters,
- digits,
- identifiers (the words of the intermediate language),
- value-names,
- measure-names (or function-names),
- property-names,
- forms of values to represent date, time, integer, and floating point numbers,
- terms that specify the proper names of objects and values of measure functions,
- operators that range over terms,
- several kinds of predicates, and
- a series of elements (E₀ - E₄) used to assert conditions on sets, objects, and relations.

The first form of element (E₀), for example, allows for such expressions as “City is Los Angeles,” and “not City is Los Angeles and not City is San Francisco.” We can quantify over sets by expressions such as “City is all,” “City is some,” “City is the,” which must be interpreted, respectively, as: each member of the set of objects must meet the conditions specified; and exactly one object must meet the conditions specified.* Membership in a set may also be qualified with respect to a cardinal number. Rule E₁ associates the membership restriction element E₀ with the name of a particular set. Rule E₂ introduces the names of properties and expressions such as, “Population is larger than 500,000,” that restrict a measure function to particular values for an object. E₃ serves to introduce the name of a relation and a specification of the codomain of the relation in terms of one or more procedures. An E₄ element specifies time, date, or both.

The next class of components are the propositions (PROP). They are the principal elements of procedures, as they specify the conditions which must be met for the storage or retrieval of information. Each proposition specifies a set, a property, or a function with restricted range. The subset of objects so specified may be further restricted by the conjunction of additional elements or procedures. The procedure imposes operations upon specific facts that have been conditionally specified by a proposition.

Relations among procedures are permitted in a proposition. For example, the question “Is the population of Los Angeles greater than the population of New York?” is represented in IL as

```
((FIND M₁ WHERE ((CITIES) AND (CITY IS LOS ANGELES))) IS*GR*- THAN
(FIND M₁ WHERE ((CITIES) AND (CITY IS NEW YORK))))
```

where M₁ is an M-name standing for the measure function “population.”

When a proposition is evaluated, it returns a “true” or “false” value and an F-name for each fact satisfying the proposition. Similarly, a procedure returns a list of F-names and operation-derived values.

The procedures statement is the highest level semantic structure in the intermediate language. Each procedure is understood to be in an inclusive “or” relationship with the others. This statement type allows us to represent the meaning of a sentence in a form similar to the disjunctive normal form of symbolic logic. Within a procedure or proposition the only connectives allowed are conjunction and denial. Disjunction occurs only between procedures. This restriction simplifies the problem of representing the semantic interpretation of a sentence and also simplifies the logic of the data management machine. There is no substantial loss of expressive power, since it has been shown that any proposition expressible in symbolic logic can be represented as a proposition in disjunctive normal form.

The data description process

Data description is a process of communicating

*The facilities to allow for quantification have not yet been added to the natural language compiler.
information about the semantic structure of a data base to the computer. This process requires the identification of general factual information, the addition of new words to the dictionary, and the addition of semantic information to dictionary entries. These processes (adding to the store of general facts and adding to the vocabulary) correspond, respectively, to the processes of axiomatic extension and definitional extension in formal systems. It is the former process which is most critical, as it is the only means of broadening the universe of discourse to include new semantic categories. The process of data description in natural language must involve a method for deriving semantic categories from the input of declarative sentences. This can be accomplished by allowing a user to introduce new words and phrases into declarative English sentences in proper defining contexts. Some of the basic declarative sentence forms which are suitable for this process are illustrated in Figure 4.

Sentence 1 illustrates one of the simplest but most important statements that one can make in describing a data base: the identification of the kinds of objects of interest. A broad universe of discourse will admit many types or sets of objects. Otherwise a universe of discourse is narrow. The natural language compiler will accept statements such as Sentence 1, will assign the syntactic category “common noun” to the term “city,” and will construct a record to contain general information about city-type objects.

Sentences 2 through 8 illustrate elementary sentence patterns that may be used to give depth as well as breadth to a characterization of a universe of discourse. Sentences 2 and 4, for example, illustrate the introduction of noun and adjectival phrases to designate descriptive (measure) functions and properties, respectively. Sentences 3 and 5, on the other hand, establish relationships between objects and between sets of objects. Sentence 3 is interpreted to mean that any member of the city set may stand in a possession relationship to one or more members of the family set. Similarly, Sentence 5 leads to a set inclusion relationship between cities and places.

Sentence 6 illustrates a most important elementary sentence pattern, one that allows us to introduce transitive verbs in such a way that the range of subjects and objects may be identified. In this simple example, the designated relation (located) can easily be associated with its domain (cities) and codomain (states). In Kellogg (1967b) we show how steps can be taken during data description to identify the converse of a relation and the logical properties (e.g., transitivity, symmetry, etc.) of relations introduced in this manner. Non-numerical values can be assigned to measure functions designated by noun phrases as shown in Sentence 7. Patterns such as Sentence 8 allows us to specify generic-specific relations among measure functions.

The meaning of new words or phrases may be defined in terms of well-formed English subset words, phrases, or sentences, as illustrated by Sentence 9. Similarly, existing words and phrases may have their meanings changed. In either case, the semantic information associated with the definiens is assigned to the definiendum. The last sentence, comparable in form to Sentence 1, illustrates the introduction of proper names for objects.

The property that a natural language compiler must have in order to accept and process sentences such as those shown in Figure 4 is the property of semantic reflexiveness. By this property we mean the ability of the compiler to accept English sentences that lead to actions increasing the range of specific facts that may be expressed in the English subset. We see that in order for a compiler to be semantically reflexive, it must be able to identify, store, and access information about semantic categories.

In a recent book, Lenneberg (1967, p. 574) concludes that: “The perception and production of language may be reduced on all levels to categorization processes . . .” and “words label categorization processes.” Of interest here is the fact that

---

(1) A city is an object.
(2) A city has a number of local government employees.
(3) Cities have families.
(4) A city may be a city.
(5) A city is a place.
(6) A city is located in a state.
(7) An English grade is A, B, C, D, or F.
(8) Distribution factors include wholesale trade and retail trade.
(9) Define population as the quantity of people.
(10) Los Angeles is a city.

Note: Doubly underlined words must be NEW, singly underlined words may be NEW.

FIGURE 4—Elementary data description sentence patterns
many of the elementary categorization processes that he goes on to discuss correspond rather closely to the semantic categories that we have found it necessary to introduce and employ in order to describe data bases in natural language.

The use of CONVERSE to describe the structure and specify the content of a portion of a data base of census information is illustrated in Figure 5. The data base derives from a standard census publication, "County and City Data Book: 1960." This data base is typical of many large formatted data bases, and it is of a size (about 600,000 facts) to require several hundred data-descriptive elements (primarily measure functions) to properly characterize its content. It contains information concerning a large number of social and economic indicators for the several political subdivisions of the United States. Sentence 1 in Figure 5A specifies political subdivisions as a principal object of interest. In this case, since we specified a singular noun form, the computer asks for the input of the plural form. (The program does not as yet contain procedures for dealing with inflectional or derivational suffixes.) We see next how a number of the important characteristics of political subdivisions may be characterized and expressed in a natural way through the use of noun phrases to describe, at varying levels of generality, some of the significant features of political subdivisions. The compiler associates each noun phrase with a corresponding measure function, and each measure function in turn is assigned a symbolic proper name beginning with the letter M (an M-name).

Sentences 2 through 13 lead to the construction of a hierarchy of measure functions. This hierarchy is stored in an explicit form that may be accessed by the compiler or by the user in furthering the data description process or in asking questions. The user knows that his input sentence has been accepted when the compiler responds as in Figure 5A with "IN," a list of fact names, or a list of M-names.

Data description may proceed in as verbose or concise a style as the user desires. The use of definitional sentences (such as Sentences 5 and 9, for example) and M-names permits concise statement of a data base description. The 13 sentences in Figure 5A produce a hierarchy of 33 descriptive measure functions (see Appendix). Each of these functions expresses a general fact about political subdivisions. It should be understood that once a measure function (or any semantic category) is specified, further definitional processes may result in a number of linkages between English words and phrases and the semantic category.

The designation of an English subset phrase or sentence is always determined from a complex process of syntactic analysis and semantic interpretation, based on the syntactic and semantic information assigned to the individual words used in the phrase or sentence. This information may have been assigned either as part of the given primitive vocabulary of English subset function words and verbs or as the result of the input data description sentences.
Several additional forms of descriptive sentences are shown in Figure 5B. Sentence 1 illustrates the introduction of adjectives. These expressions map into properties that are true of discourse. The semantic structure of the data base is then extended to include additional forms of objects—cities, counties, states, and places. A compound sentence is used to express an equivalence relationship between political subdivisions and places and a set inclusion relationship between these classes and the classes of cities, counties, and states. Each general fact that applies to political subdivisions and places now applies also to cities, counties, and states. In addition, cities and counties are further distinguished in Sentences 3 through 7.

Figure 5C illustrates several of the sentence forms available for the specification of the proper names of objects. Just as the first step in describing a data base is to name one or more kinds of objects of interest, the first step in specifying specific factual information is to assign proper names to one or more objects. We note that proper names do not have to designate uniquely. The ambiguity of reference for the proper names, Los Angeles, Santa Barbara, and New York, is easily resolved later on, wherever they are used in appropriate contexts in English subset sentences.

The English subset has now been extended to the point where a wide variety of facts about cities, counties, states, and places may be introduced and specified through the use of additional declarative sentences. Several of the admissible syntactic patterns and variations in style are illustrated in Figure 5D. This figure clearly indicates how specific facts may be expressed in either a verbose style or in a highly concise manner reminiscent of the artificial languages used in current data management systems. We believe a natural language subset should admit the concise patterns of expression shown in Sentences 5 through 10, as well as sentences such as 1 through 4.

The compiler response to Sentence 1 illustrates a typical feedback message to the user. In this case an ambiguous input sentence is detected. Re-phrasing the sentence as in (2) results in acceptable input.

**Figure 5D—Specific information**
Extending the data description to encompass new relationships

Figure 6 illustrates a user-computer dialogue in which the range of objects and relations is extended beyond that given earlier in Figure 5. In this case, we are extending the universe of discourse to encompass information similar to that found in airline flight tables.

A new class of objects, "flights," is introduced, and a member of that class, "PA-25," is specified. Several adjectival modifiers are specified and "O'Hare" is then defined as a place. Sentences 5 and 6 illustrate the introduction of new transitive verbs into the English subset—in this case, verbs designating departure and arrival relations. The input of these sentences results in the generation of two new general facts, F21 and F22, that specify the relation names, the domains of the relations, and their codomains. Next, two simple definitional statements are used to establish the equivalence of the singular and plural forms of the two verbs. With the input of this information, the compiler is now able to process and translate questions of the form shown into IL search procedures.

Notice that the expressions "National" and "Pan Am" designate properties which may be true or false for specific flights. This is the simplest means of distinguishing the different kinds of flights. We could have used sentences that would have led to a further increase in the semantic structure. For example, we could have typed in sentences indicating that "Pan Am" and "National" are "airlines" and then the fact that "airlines have flights." This is a simple illustration of an important principle: The level of detail of the data base description is variable and under the control of the person describing the data base.

In the census data base, for example, it is possible to define persons as objects and to add the fact that "cities have people." Then, instead of asking a question such as "What is the population of Los Angeles?" we could ask the question, "What is the number of people in the city of Los Angeles?" The computer would then be directed to count the number of "people" objects in the city.

Getting back to our departure and arrival relations, we can extend the scope of the codomain of an already defined relation such as "depart" to include the notion of departure time. By typing in "a flight departs at a time," a third codomain argument would be added to the two existing arguments (from place, for place) and the general fact for "depart" would be changed accordingly. We could then ask questions such as "What flights depart from Los Angeles for O'Hare before 7:30 a.m.?" If we wish to ask a question such as "What is the departure time for PA-25?" we can definitionally extend the English subset by typing in "Define departure time as departs at a time." This would establish a nominal to designate the third argument of the codomain of the departure relation and thus the set of departure times.

Deriving semantic structures from syntactic structures

The process of compiling sentences into procedures consists of dictionary lookup, syntax recognition, and three interrelated stages of semantic interpretation: semantic resolution, element construction, and procedure construction. The syntactic recognition process controls the application of semantic interpretation rules in accordance with the grammatical relations that are recognized during syntactic analysis. Just as syntax recognition is a process of recursive decomposition of a sentence into its syntactic categories and words, semantic interpretation can be conceived of as the recursive composition of elementary semantic values and categories into semantic structures. The satisfactory implementation of semantic interpretation processes requires not only a set of semantic interpretation rules but a dictionary structure that is rich enough to represent the complex associations obtaining between English words and the semantic categories and values they designate. Specifically, provision must be made for an English term to take on a number of meanings with respect to the contents of a data base.

The principal information structure for accomplishing sentence-to-procedure translation is the component. Lexical components reside in the dictionary, while derived components are the result of the application of rules to lexical components or to other derived components.

- A lexical component consists of: a word or idiom, syntactic and semantic categories, and a lexical interpretation list.
- A derived component consists of: a syntactic structure, and a derived interpretation list.
- An interpretation list consists of one or more interpretations.
Each interpretation represents one specific meaning of a word or syntactic structure and consists of:

- a feature list (F),
- a selection restriction list (SELR),
  and sometimes:
- an element list (E), and
- a procedure list (P).

An entry in our dictionary consists of a lexical component and additional associated information such as pointers to general or specific facts. This information is stored in LISP as a property list associated with the entry word or idiom.

During dictionary lookup, the hash coding system built into LISP for recognizing primitive symbols (atoms) is used to locate the entries for words. At this time digit sequences are recognized and assigned interpretations as integer or floating point values, as appropriate. The original input string is rewritten to include recognized syntactic categories.

Syntax analysis proceeds in a top-to-bottom, left-to-right manner. List structures are used to represent recognized syntactic elements in derived components. In general, as a pair of syntactic elements is recognized, one or more semantic rules are applied to derive an interpretation list for that pair.

We have discussed the rationale behind the use of interpretation list structures elsewhere (see Kellogg, 1967a). Briefly, each interpretation is an individual meaning-bearing element that may be pointed to by any number of words or syntactic structures. It always consists of at least a list of semantic (and sometimes syntactic) features and a selection restriction. These constructs generally perform the same functions as the corresponding concepts in the theories of Katz (1966) and Chomsky (1965). By themselves, features and selection restrictions may represent the sense of an English word and the contexts or environments in which that sense can correctly occur. For example, the meanings of certain prepositions, adverbs, and determiners in the English subset are adequately represented in this manner. However, if an interpretation must also designate semantic categories and values, an element list is required. Finally, as elements are combined to form parts of a procedure, they must be moved to a procedure list.

Since only a part of the burden of semantic interpretation is placed on the use of semantic features (their use is restricted to indications of simple class membership, class inclusion, agreement, etc.), we have not found it difficult in practice to assign features during the data description process. Our experience in using interpretation lists confirms their value as a good solution to the problem of distributing semantic information between the dictionary and rules. Imposing a heavier semantic load on features can lead to a number of difficulties (see Bolinger, 1965), while dictionary entries of significantly simpler structure can result in the need for a very much larger set of semantic interpretation rules.

Semantic resolution rules apply to components that stand in a specific grammatical relationship to one another. Their function can be understood by considering the application of the following rule:

\[
\text{SAT}[F, \text{SELR}] \rightarrow (\text{UNION}[F_1, F_2], \text{SELR}_2, \text{UNION}[E_1, E_2])
\]

for the following components:

\[
\text{ALPHA}_1 \rightarrow \begin{cases} \text{and} (x_1, x_3) & \text{if } x_3 < x_1 \\ x_1 & \text{if } x_3 \geq x_1 \end{cases}
\]

\[
\text{BETA}_2 \rightarrow \begin{cases} \text{or} (x_4, x_5) & \text{if } x_4 < x_5 \\ x_4 & \text{if } x_4 \geq x_5 \end{cases}
\]

The rule applies if the word “alpha” grammatically modifies the word “beta” and the left half or pattern part of the rule evaluates to “true.” These rules have a “pattern \rightarrow operation” format. The pattern part of a rule consists of predicates that return either true or false values. The right half consists of symbols and operations over symbols that construct a derived interpretation. In the above example, the first component has two interpretations, each having a list of features (enclosed in parentheses), a selection restriction (a Boolean function of features enclosed in angle brackets) and an element list (enclosed in square brackets). The “beta” component has three interpretations (selector restrictions are not shown in this case in order to simplify the example).

The rule applies to every possible combination of “alpha” and “beta” interpretations, and if the left half of the rule returned “true” in each case, then six derived interpretations would result. In this example, only three interpretations are allowed by the resolution rule.
The derived component is:

The semantic rule has resolved a possible sixfold ambiguity into a threefold ambiguity. The other three possible interpretations were rejected by the failure of the "SAT" predicate to match the selection restriction of an "alpha" interpretation (SELR), to the features list of a "beta" interpretation (Fβ).

In general, resolution rules allow us to check for compatibility or agreement between interpretations and to map pairs of compatible interpretations into derived interpretations. Element lists, when they occur, are usually combined by a simple union operation as shown.

Two additional sets of rules must be applied to derived interpretations, in sequence, in order to construct well-formed semantic structures. Element construction rules combine the symbols in an element list into well-formed elements as specified by the formation rules E0 - E4 in Figure 3. Similarly, procedure construction rules move elements to the procedure list (P-list) in accordance with intermediate language formation rules, insuring the production of only well-formed procedures.

The feature, selection restriction, element, and procedure lists, in conjunction with the three kinds of semantic interpretation rules, constitute a combinatorial apparatus of considerable power and flexibility. Our experience so far indicates that such a system of information structures and rules is adequate for the processing of a fairly wide class of English sentences, and furthermore, that a translation procedure based on such structure and rules is reasonably efficient. (The version of the CONVERSE system in current use on the Q-22 Time-Sharing System at SDC typically requires from less than one second to as much as three seconds of computer processing time to effect sentence-to-procedure translation.)

An example of question-to-procedure translation

Figure 7 illustrates the syntactic structure that is assigned by the compiler for the question "Which Pan Am flights that are economy class depart for O'Hare from the City of Los Angeles?"

Figures 8 and 9 illustrate the syntax rules for this question, the applicable resolution (CR), element (ER), and procedure (PR) rules, and the actions of semantic rules that result in the construction of a well-formed procedure.

Rules CR, and ER, are applied to each adjective phrase to test for the features "SETP" and "PTER." Where these conditions are met ("Pan

FIGURE 8—Syntactic and semantic rules applied to the sample question

FIGURE 7—Compiler-assigned syntactic structure for a sample question
Am, "Economy Class") we substitute the feature “SETP” for “SETP” in the derived feature list. This indicates that a phrase has been identified as designating a property and rule ER, is applied to test for the designation of a single unique property (specified by its P-name). A P-rule is first applied in conjunction with syntax rule SR. Each P-rule has as its pattern part a set membership test. For rule PR, a Pp element or P-name found in an element list is replaced by the name of the IL formation rule (E2) that dominates it and the P-name is added to the P-list. This procedure allows us to keep track of the level of the semantic categories as well as to combine them, as necessary, in the P-list.

Rules CR, ER, and PR are used to test the object and subject of the transitive verb and to construct a list of proper arguments for the designated relation. CR combines the several interpretations of prepositional phrases into a single list. The CDSAT predicate in Rule CR applies the selection restrictions of the verb “depart” to the interpretation list formed by Rule CR. The selection restriction for this verb consists of a pointer to the general fact representing the relational meaning of the verb. In this case, the general fact has the form: (flight.X) (departs) (place.Y) (place.Z) and the result of the first application of CR and ER is the element list ( (departs), ( ), (place is O'Hare)), where the empty list stands for the missing argument: (place.Y). The second application of these rules (to the interpretation of “from the city of Los Angeles”) results in the filling in of the missing argument. Finally, Rule PR appends the relation name and its codomain arguments to the end of the P-list. The last steps in compilation include a test for subject-verb agreement and the addition of a “print” operation to the P-list.

Several properties of the semantic rules are of significance. We notice that with few exceptions (i.e., SETP, PTER) the rules are formulated independently of specific features. Were this not the case, we would run the risk of requiring new rules every time we described or modified the semantic structure of a universe of discourse. Secondly, only a small number of predicates and operations are required to formulate the rules (only a few more than shown here are required in the complete system). Thirdly, only a relatively modest number of semantic rules are required (the present system requires less than three times as many semantic rules as are shown in Figure 9).

Finally, though not illustrated here, the semantic rules always correctly distribute ambiguous or multiple requests to separate procedures.

The data management machine

The data management machine is at present implemented as a series of LISP procedures. Though only a modest number of facts can be handled within LISP, the information structures used are both simple and general enough to form the basis for a large-scale storage system.

Three kinds of facts are allowed: A simple fact expresses a relation among specific objects. For example, the list:

```
((departs), (flights.PA-25), (city.Los Angeles), (place.O'Hare), (time.1730))
```

represents the fact that flight PA-25 departs from Los Angeles for O'Hare at 5:30 p.m. Many requests can be answered by consulting compound facts, in which individual objects are characterized by a vector of elements such as:

```
((cities), (city.Los Angeles), P, (M, 40, 000), (date.680427))
```

This is an example of a compound fact that states
that on the 27th of April 1968 property P was true and function M, had a value of 40,000 for the city of Los Angeles. Only one date and/or time value is allowed for each compound fact and a new compound fact is constructed for each description of an object at a different point in time. A compound fact becomes complex when the object it describes enters into a relationship with other objects. An example is:

\[(\text{flights}), (\text{flight.PA-25}), (\text{departs}, \text{5' FlO})] \]

This fact links all of the information about flight PA-25 to those members of the "departs" relation desired degree of complexity.

The data management machine offers a significant increase in the range of possible universes of discourse as contrasted with, for example, the BASEBALL system (Green, 1963) which was limited to the description of specific baseball games and the SIR system (Raphael, 1964), which was limited to the recognition of a small number of relationships.

*It is readily seen that simple and compound facts express atomic and molecular propositions respectively (see Travis, 1963). A complex fact represents a molecular proposition that associates the object descriptions that participate in the R-image for a specified object and relation.*

**FIGURE 10—Elementary question patterns**

**FIGURE 11—Compound requests-I**

**FIGURE 12—Compound requests-II**
FIGURE 13—The use of definitions in phrasing complex requests

Examples of data base interrogation

Figures 10 through 16 illustrate some of the interrogation patterns acceptable to the CONVERSE system at present. Figure 10 illustrates some elementary question patterns. Question 1 represents perhaps the simplest, most general (and least interesting) question we can ask, as it translates into a request consisting only of a
"print" operation. Execution of a file-searching procedure of this kind would result in a listing of all the values in the complete data base. Similarly, for Question 2, a listing of all of the population values associated with objects in the data base would result. Question 3 compiles into two separate requests. The answer to the first request (ANSWER1) simply confirms the fact that Los Angeles is the name of a specific county since we have stored no additional specific data for the county of Los Angeles. ANSWER2, however, yields the complete record of specific information for the city of Los Angeles. The last question illustrates a simple phrasing of a set membership request.

Figure 11 illustrates some of the patterns for combining and coordinating terms into phrases and sentences. Sentence 1, for example, illustrates how two coordinated proper nouns may be post-modified by a common noun. The proper nouns are distributed into two separate requests, since a single request calling for the population of a "Los Angeles and New York" object would be absurd. Similarly, Sentence 2 results in two requests where the common nouns are both pre-modified by the same adjective. Sentence 3 represents a somewhat more involved coordination pattern as two noun phrases designating generic measures are post-modified by the same prepositional phrase. A single, rather lengthy, request results for this question type. In Sentence 4 two coordinated common nouns are first postmodified by the same adjective. Sentence 3 represents a somewhat more involved coordination pattern as two noun phrases designating generic measures are post-modified by the same prepositional phrase. A single, rather lengthy, request results for this question type. In Sentence 4 two coordinated common nouns are first postmodified.

FIGURE 15—Modifying the "meaning" of terms...
largest cities in population" are identified by finding in turn the city with the largest population value and then the city with the next largest population value.

The many devices for compounding and coordination in natural language allow us, in a data management context, to quite easily phrase questions that compile either into complex requests or into a large number of individual requests. As an illustration of the latter case, consider the question in Figure 13: "For the smoggy high-income cities what is the age-income value-range?" In order to derive the necessary data from the data base, twenty requests in the intermediate language result from the compilation of this question. Some of the potential of, and justification for, research work in natural language data processing is evident from this example. The user's alternative to the phrasing of his question in English is to type, in some artificial query language, twenty requests comparable to those shown, or some highly complex nested Boolean statement that is the equivalent of these requests. The computer response (in brief form) allows us to see quickly that requests 1, 15, and 19 are satisfied by the city of New York; 2, 14, and 20 are satisfied by Los Angeles, and that requests 8 and 10 are satisfied by both the city of New York and the city of Los Angeles (future versions of the system will generate answer statements from requests and their non-nil responses). Definitional sentences to specify appropriate concepts of high income, distribution of people according to age and income, and a range of percentage values of interest to the user are shown at the bottom of Figure 13.

Figure 14 illustrates several kinds of feedback messages that may result from user interrogation of the semantic structure of the universe of discourse. Suppose, for example, that a user wants to know the normative descriptive phrase associated with a specific M-name. He might first try a question such as Question 1 which leads, as we see, to an "undefined word" feedback response. Similarly, if he rephrases the question as in Sentence 2, then all words are defined but are not put together in an acceptable syntactic structure. Finally, if he phrases his request as in Sentence 3, he gets a simple direct response as a procedure specifying the normative description phrase for the M-name.

If he is concerned about the referent for a particular noun phrase, a "what about" question, as expressed in Sentence 4, will generate a separate procedure for each referred object. The response to Question 5 indicates that the noun phrase "medium family income" is well formed and designates, in this case, a specific measure function. Similarly in Sentence 5, we see that the phrase "pop (population) factors" designates a list of specific measure functions. In Sentence 7, we see that the constituent "percentage of people," while syntactically well formed, does not evaluate to an object, function, or property, in the intermediate language. Hence, its meaning is undefined and an appropriate feedback message results. However, we may rephrase this question as in Sentence 8, in order to find all measures that are directly or indirectly linked to the phrase.

Figure 15 illustrates one means of changing and extending the meaning of English subset terms. Sentence 1 illustrates the use of "large" as this term was specified during data description. The second sentence in Figure 15 illustrates a definitional extension that adds to the set of interpretations assigned to an English subset constituent. The notion of "large" is extended to include objects that have a "population" over a certain value and a "quantity of manufacturing establishments" larger than a certain value. Question 3, "What are the large, smoggy Eastern cities?" then compiles into two requests as shown. In this case the answers for the first and second requests are the names of the same city. The remainder of the sentences in this figure illustrate how the original interpretation of the word "large" can be replaced by the second interpretation. The ability to change the number and kinds of interpretations associated with a constituent is an essential function in providing a user with a flexible and user-extendable English subset.

Figure 16 illustrates a brief sketch of the description and interrogation of a fragment of a data base concerning information about secondary school students.

CONCLUSION

We have realized a prototype on-line system for describing, updating, and interrogating data bases of diverse content and structure through the use of ordinary English sentences.
We cannot hope to formalize and implement a complete English grammar in the near future; instead, we face a tradeoff between range of expression and reasonable computer processing times. Therefore, in constructing a grammar for an online English subset it is as important to avoid some areas of syntax as it is to recognize others.

By composing derived interpretations from lexical interpretations, under the control of syntax recognition, a process of semantic interpretation has been realized to construct the designations of noun phrases, sentences, and embedded clauses. The intermediate language constitutes a database-oriented "deep" structure. It comprises a framework into which new kinds of objects, predicates and terms may be introduced to represent the semantic structures of a very wide range of possible universes of discourse. In a similar way the data management machine's specific fact structures present a framework for the storage of the extension of a universe of discourse.

Several properties of the CONVERSE natural language compiler are of special importance.

- **Ambiguity resolution.** The semantic interpretation process eliminates interpretations that are not admissible according to selectional feature restrictions and general facts. Syntactic ambiguity is resolved by terminating syntactic structures that do not lead to acceptable semantic structures.

- **User feedback.** Undefined word, structure, meaning, and procedure statements are particularly useful in guiding a user to an awareness of the limits of the English subset and in enabling him to extend the English subset to meet his needs.

- **Semantic reflexivity.** This property allows a user not only to introduce new universes of discourse but, through definitional sentences, to increase the vocabulary and paraphrase facilities of the English subset. It allows him to describe a database at either a superficial or detailed semantic structure level.

- **On-line responsiveness.** In the present version of CONVERSE a user must typically wait from less than five seconds to as much as one minute for sentence-to-procedure translation. The system is presently being converted for use on the IBM 360 Time-Sharing Systems at SDC. This new program promises to be at least one order of magnitude faster than the present one.

In conclusion, we believe that a natural language compiler should be judged on its merits, not with respect to the linguistic possibilities inherent in man-to-man communication, but with respect to existing means of man-to-machine communication. From this viewpoint, we are convinced that natural language compilers will eventually come into widespread use, for the same reasons that conventional compilers are already being widely used. The gain in convenience and expressive power will more than offset the expense of the required computer processing time.

ACKNOWLEDGMENT
Work on CONVERSE has been carried out within the Natural Language Research Program at SDC with the helpful and enthusiastic support of Robert F. Simmons. Carter C. Revard and Larry E. Travis have supplied valuable comments on a first draft of this paper.

APPENDIX

PRINT MEASURES . PROPERTIES .

| M1 | NIL |
| M3 | (PRODUCTION FACTORS) (M30 M31 M32) |
| M4 | (DISTRIBUTION FACTORS) (M22 M23) |
| M5 | (POPULATION FACTORS) (M7 M8 M9 M10) |
| M6 | (LOCAL GOVERNMENT FACTORS) (M33 M34) |
| M2 | (STATISTICAL INDICATORS) (M3 M4 M5 M6) |
| M7 | (DISTRIBUTION OF PEOPLE BY AGE) (M11 M12 M13 M14) |
| M8 | (DISTRIBUTION OF PEOPLE BY INCOME) (M15 M16 M17 M18) |
| M9 | (DISTRIBUTION OF PEOPLE BY EDUCATION) (M19 M20 M21) |
| M10 | (POPULATION) |
| M11 | (MEDIAN AGE OF PEOPLE) |
| M12 | (PERCENTAGE OF PEOPLE UNDER 5 YEARS OF AGE) |
| M13 | (PERCENTAGE OF PEOPLE OVER 21 YEARS OF AGE) |
| M14 | (PERCENTAGE OF PEOPLE OVER 65 YEARS OF AGE) |
| M15 | (MEDIAN FAMILY INCOME) |
| M16 | (PERCENTAGE OF FAMILY INCOME UNDER 3000) |
| M17 | (PERCENTAGE OF FAMILY INCOME OVER 10000) |
M18 (AGGREGATE FAMILY INCOME IN MILLIONS OF DOLLARS)
M19 (PERCENTAGE OF PEOPLE OVER 25 WITH LESS THAN 5 YEARS OF SCHOOLING)
M20 (PERCENTAGE OF PEOPLE OVER 25 WITH MORE THAN 12 YEARS OF SCHOOLING)
M21 (MEDIAN NUMBER OF SCHOOL YEARS COMPLETED)
M22 (RETAIL TRADE) (M24 M25 M26)
M23 (WHOLESALE TRADE) (M27 M28 M29)
M24 (NUMBER OF RETAIL TRADE ESTABLISHMENTS)
M25 (NUMBER OF RETAIL TRADE EMPLOYEES)
M26 (RETAIL TRADE YEARLY PAYROLL IN THOUSANDS OF DOLLARS)
M27 (NUMBER OF WHOLESALE TRADE ESTABLISHMENTS)
M28 (NUMBER OF WHOLESALE TRADE EMPLOYEES)
M29 (WHOLESALE TRADE YEARLY PAYROLL IN THOUSANDS OF DOLLARS)
M30 (NUMBER OF MANUFACTURING ESTABLISHMENTS)
M31 (NUMBER OF MANUFACTURING EMPLOYEES)
M32 (MANUFACTURING PAYROLL IN THOUSANDS OF DOLLARS)
M33 (NUMBER OF LOCAL GOVERNMENT EMPLOYEES)
M34 (LOCAL GOVERNMENT PAYROLL IN THOUSANDS OF DOLLARS)
M35 (TEMPERATURE FACTORS) (M39 M40 M41 M42)
M36 (MEAN ANNUAL PRECIPITATION)
M37 (MEAN ANNUAL HOURLY WIND VELOCITY)
M38 (MEAN JAN TEMPERATURE)
M39 (MEAN JULY TEMPERATURE)
M40 (HIGHEST TEMPERATURE)
M41 (LOWEST TEMPERATURE)
M42 (FARM LAND IN THOUSANDS OF ACRES)
M43 (NUMBER OF FARMS)
M44 (AGRICULTURAL FACTORS) (M44 M45)
P1 (SMOGGY)
P2 (WARM)
P3 (COLD)
P4 (EASTERN)
P5 (WESTERN)
P6 (LARGE)
P7 (SMALL)

REFERENCES

D G BOBROW J B FRASER M R QUILLIAN
Automated language processing

H G BOHNERT P O BACKER
Automatic English-to-logic translation in a simplified model

D BOLINGER
The automation of meaning
Language (October–December 1965) Vol 41 No 4 555–573

E BOOK
The LISP version of the meta compiler
SDC document TM-2710/330/00 November 1965

Bureau of the census
County and City Data Book 1962

U S Department of Commerce Washington D C 1962

R CARNAP
Meaning and necessity
2d ed University of Chicago Press Chicago 1956

T E CHEATHAM JR S WARSHALL
Translation of retrieval requests couched in a semiformal English-like language
Communications of the ACM 1962 5(1) 34–39

N CHOMSKY
Aspects of the theory of syntax
Cambridge: The MIT Press 1965

W D CLIMENSON
File organization and search techniques
In Cuadra C A (Ed) Annual Review of Information Science and Technology New York John Wiley and Sons 1966

W A COZIER W C DENNIS
QUIP users’ manual SDC document TM-2711/000/01 February 1966

V E GIULIANO
Comments (on Simmons (1965))
Communications of the ACM 1965 8(1) 69–70

B F GREEN JR A K WOLF C CHOMSKY
K LAUGHERY
Baseball an automatic questions answerer

J J KATZ
The philosophy of language
New York Harper and Row 1966

C KELLOGG
An approach to the on-line interrogation of structural files of facts using natural language
SDC document SP-2431/000/00 29 April 1966

C KELLOGG
On-line translation of natural language questions into artificial
language queries
Information Retrieval (in press) (a)

C KELLOGG
CONVERSE—A system for the on-line description and retrieval of structured data using natural language
Proceedings of the FID/IFIP Conference on Mechanized Information Storage Retrieval and Dissemination
Rome Italy June 1967 North Holland Pub Co (in press) (b)

C KELLOGG
Designing artificial languages for information storage and retrieval In Borko H (Ed)
Automated Language Processing
New York John Wiley and Sons 1967

C KELLOGG
Data management in ordinary English: examples
SDC document TM-3919/000/00 3 May 1968

E H LENNEBERG
Biological Foundations of Language
John Wiley and Sons Inc New York 1967

J MINKER J SABLE
File organization and data management

In Cuadra C A (Ed) Annual Review of Information Science and Technology New York John Wiley and Sons 1967

B RAPHAEL
SIR a computer program for semantic information retrieval
AFIPS Conference Proceedings

R F SIMMONS
Answering English questions by computer: A survey
Communications of the ACM
1965 8(1) 53-70

R F SIMMONS
Automatic language processing
In Cuadra C A (Ed) Annual Review of Information Science and Technology
New York John Wiley and Sons 1966

L TRAVIS
Analytic information retrieval
In Garvin P L (Ed) Natural Language and the Computer