DEACON: DIRECT ENGLISH ACCESS AND CONTROL *

James A. Craig
Susan C. Berezner
Homer C. Carney
Christopher R. Longyear

General Electric Company, Santa Barbara, California

INTRODUCTION

The extensive syntactic ambiguity inherent in natural language has been convincingly shown by such systems as the Harvard syntactic analyzer. Furthermore, no semantic techniques are in prospect for satisfactory resolution of this ambiguity by computer. In contrast, well-developed semantic techniques exist for formal languages.

In an accompanying paper, Thompson defines a formal language and a technique for determining the meaning of sentences in that language. The semantic technique is to use interpretation rules which define actions (or sequences of actions) involving the objects of an environment. An environment is defined as a finite set of categories of computer memory structures. Thompson hypothesizes that English essentially becomes a formal language as defined if its subject matter is limited to “material whose interrelationships are specifiable in a limited number of precisely structured categories [memory structures].”

The DEACON system constitutes a test (and, we feel, a confirmation) of that hypothesis. The environment, in this application, consists of “ring” structures in which data is stored as it is introduced into the system. The interpretation rules are computer programs that perform various operations on the ring structures.

Because these programs are written in terms of structural categories (independent of content), the interpretation rules apply to any subject matter that is stored in these categories. Each interpretation rule is associated with a rule of a context-sensitive phrase structure grammar. A combination phrase structure rule/interpretation rule determines the meaning of the phrase that is formed by applying the rule. A sequence of rules applied by a parsing routine determines the meaning of an English sentence, phrase by phrase, in accordance with the subject matter of the data base. Simmons has called this “parsing directly into a data structure.”

* The DEACON project is primarily supported by the Rome Air Development Center under contract AF30(602) 4272 and also by the Research and Development Center of the General Electric Company.

† Not only did Thompson develop the theoretical basis for DEACON, but he also directed and participated in all aspects of its application before joining the staff of the California Institute of Technology.

‡ This is the major advance of DEACON over Green’s BASEBALL and Lindsay’s SAD SAM. These earlier systems provided valuable background for DEACON, and discussions with Lindsay on DEACON itself were extremely helpful.
The question of subject-matter limitations (which are necessary in order to use formal-language semantic techniques on English) is crucial to the effectiveness of a DEACON system. The severity of the limitation depends inversely on the adequacy of the memory structures that are used. Among the more advanced structures yet devised are ring structures, such as those used by Sutherland in SKETCHPAD. The particular ring structures now used in DEACON (devised by Thompson, with contributions by R. Donald Freeman, Jr.) are described later in this paper.

Ring structures are adequate for storing a wide range of richly interrelated data that is pertinent to such functions as intelligence analysis, management planning, and decision making. Typical of these functions are resource allocation problems, in which the pertinent data is an inventory of the resources, their characteristics, and their interrelations. This type of data is specifiable in ring structures.

It is for such management functions that DEACON is being developed. DEACON as a management information system has the following characteristics: Any particular management staff works with a private data base relevant to its own operations, rather than with a universal or pre-established data bank. A user puts data into the system by typing appropriate English statements at a teletype in his normal working area. Similarly, he elicits data by typing English queries. Whether inputting data or asking questions, the user need not know how the data is stored. The interpretation rules automatically relate English statements to the data structures stored in computer memory. This permits the user to concentrate on his problem rather than irrelevant detail.

The use appropriate for a DEACON system differs in a number of important respects from uses which might be thought appropriate for large, fixed-format systems. Characterization of human informational processes suggests the theoretical inadequacy of fixed-format systems in any real application and also predicts that the most efficient informational processes occur only within a small informational community concerned with shared subject matter in a rapidly changing context. The intended use of a DEACON-type system based on augmenting human informational processes is discussed more fully in references 7 to 11. These references present both the philosophy of informational processes and illustrations of the anticipated use of DEACON-type systems.

This paper is intended to illustrate the application of the theory presented in Reference 2. It describes the use of interpretation rules by the DEACON system in analyzing and responding to English sentences on the basis of stored data. The following section shows the functioning of a typical rule in abstraction and then illustrates its use along with others in analyzing a sample sentence. Ring structures and how they are built (data input rules) are then described, followed by a description of the use of a “verb table” in analyzing certain types of sentences. Next is a description of the parser that applies the grammar rules, followed by a brief description of the hardware configuration used and then some conclusions. An appendix describes some more grammar rules and lists sample sentences and corresponding system responses.

**INTERPRETATION RULES IN SENTENCE ANALYSIS**

To analyze a sentence, the DEACON system must be able to recognize the strings of characters that form the sentence. To accomplish this, a dictionary of vocabulary terms is built up from definitions typed in by the user (currently in an interactive mode of operation but using formatted statements). A vocabulary term may be a word (a string of characters between blank characters in a sentence) or an idiom (a string of two or more words, such as San Francisco).

Because of the nature of the subject matter of a DEACON system, most of the vocabulary terms are those normally considered to denote objects, their characteristics and their interrelations. In DEACON, however, these terms denote structures in the data base. The examples used in this paper assume as subject matter a simulated army environment. In the sample data base, or subject matter characterization, the term BATTALION denotes a ring containing data about battalions, and the term 638TH denotes a ring containing data about a particular battalion named the 638TH. The ring denoted by 638TH may also be denoted by a more descriptive name, such as BLUE EAGLES. Or a given vocabulary term may denote more than one ring.

The rings mentioned above are called the data referents, or referent rings, of the vocabulary terms that refer to them. The referent rings may be thought of as doors to the data, with specific data...
entries consisting of connective rings intersecting appropriate referent rings, as shown later.

The terms that denote rings are called referent terms, and usually are assigned part of speech R, for ring, or V, for verb.* Function words, such as prepositions and articles, normally do not denote rings. In some areas of the system, these words are treated as a class, and are identified with part of speech F. In rules of grammar, however, each function word is its own part of speech.

Other parts of speech are: N for number; X for pronoun; T for time; S for sentence. These will be discussed as they arise.

The definition of a vocabulary term consists of a part of speech, grammatical information, and (for denotational terms) a link to a referent ring in the data base.

When a sentence is typed into the system, a dictionary lookup routine finds all the pre-defined vocabulary terms from the sentence. From each voc-

* These parts of speech indicate something about the types of structures used for storing referent data of vocabulary terms that have these parts of speech. These parts of speech therefore function as both syntactic word classes and semantic categories. In this paper, they are called either “part of speech” or “category.”

cabulary term found, the system forms an initial phrase. In this paper, therefore, “phrase” applies to single words as well as to syntactic constructions. The DEACON representation of a phrase also includes its part of speech, grammatical information, and a link to a referent ring.

A parsing routine applies rules from the system’s grammar to combinations of initial phrases to form syntactic constructions, or intermediate phrases. Each rule consists of two parts—the syntactic, or phrase structure, part and the semantic, or interpretation, part.

The phrase structure part of a rule specifies the part of speech symbols (categories), grammatical characteristics, and positional relations that its input phrases must have. Similarly, it specifies the part of speech and grammatical characteristics of its output phrase. For example, phrase structure Rule 5 states that two adjacent R-phrases (which may be initial or intermediate phrases, each with part of speech R) may be combined to form an intermediate R-phrase. The syntactic construction of this intermediate R-phrase is shown in tree form in Part A of Fig. 1. The fact that this part of the rule deals with part of speech symbols (not with the associated vo-

---

**LEGEND:**

- Referent ring.
- Connective ring.

**Figure 1.** The functioning of a typical DEACON grammar rule.
The vocabulary term is emphasized by writing the phrase structure rule as $R + R \rightarrow R$, where “+” indicates concatenation. The specific grammatical checks (for number, case, etc.) are not shown.

After phrase structure Rule 5 is successfully applied, interpretation Rule 5 is applied. The interpretation rule first determines whether the phrase being formed is meaningful in the data base by looking for a certain type (or types) of connective ring that may intersect the referent rings of the input R-phrases. In our example, a two-link connective ring satisfies the rule’s conditions, as shown in Part B of Fig. 1. Therefore, the rule establishes a link in the output phrase to a referent ring which, in this case,* is the referent ring of the first input R-phrase (shown as $R_i$). To show the correspondence of the referent ring of the output phrase to the referent rings of the input phrases, the rule may be written as $R_i + R_j \rightarrow R_k$.

If no appropriate connective ring is found, the rule application is aborted as though the phrase were ungrammatical. (Actually, the phrase is marked “VACUOUS DESCRIPTION” and is later typed out as a response if no alternative analyses result in a different response.) Eliminating such constructions that are “grammatical but meaningless for the subject matter of the data base” is a primary technique for controlling ambiguity in the system.

The following sentence illustrates a specific application of Rule 5 and other rules:

**WHO IS COMMANDER OF THE 638TH BATTALION?**

The relevant dictionary entries, data base entries, and cross references are shown in Fig. 2. The dictionary lookup produces the following internal representation of the sentence:

<table>
<thead>
<tr>
<th>Spelling</th>
<th>Part of Speech (Category)</th>
<th>Reference to Data Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>638TH</td>
<td>R (RING WORD)</td>
<td></td>
</tr>
<tr>
<td>BATTALION</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>COMMANDER</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>OF</td>
<td>F (FUNCTION WORD) &quot;OF&quot;</td>
<td></td>
</tr>
<tr>
<td>JONATHAN M. PARKER</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>THE</td>
<td>F &quot;THE&quot;</td>
<td></td>
</tr>
<tr>
<td>WHO IS</td>
<td>F &quot;WHO IS&quot;</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>F ?</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Sample dictionary and data base entries, and their cross references.

The parser, matching grammar rules with combinations of initial-phrase parts of speech in the internal representation of the sentence, finds that Rule 5 (Fig. 1) applies to the string “638TH BATTALION.” Its application is illustrated in Fig. 3A. Since the application is successful, the internal representation of the sentence is rewritten with the output phrase of the rule replacing the two input phrases (see Fig. 3B). In this case, the effect is the same as if the phrase BATTALION had been dropped from the original representation (after verifying that the 638TH is a battalion).

Next, the parser matches its rules against the new sentence representation. It finds Rule 11, THE + R → R (see Fig. 4). This rule in effect drops the article THE. It also illustrates the use of a function word as its own part of speech. Again, the sentence representation is rewritten.

Figure 5 shows the application of Rule 8, $R_1 + OF + R_2 \rightarrow R_3$, to COMMANDER OF 638TH. This rule introduces a new feature, i.e., that the output phrase refers to a referent ring not mentioned in the input sentence. Through the rules applied thus far, the string COMMANDER OF THE 638TH BATTALION has been replaced by its equivalent, JONATHAN M. PARKER.

The application of a final rule, WHO IS + R + ? → S, forms a sentential phrase and sets up the

* Other rules that have two R-phrases as constituents but which produce different output phrases are discussed in the Appendix.
answer to the question (see Fig. 6). The system is ready to accept another message after the output routine types the answer:

WHO IS COMMANDER OF THE 638TH BATTALION?

JONATHAN M. PARKER

The overall sentence analysis is shown in Fig. 7. Conceptually, the system checked the data base to ensure that the unit called 638TH is a battalion, found the portion of the 638TH data record that designated its commander, and typed the associated name as the response to the sentence.

This illustration shows only one analysis of the sample sentence. However, if either syntactic or referent ring ambiguity causes alternative interpretations of the sentence, each corresponding answer is typed out. Techniques for dealing with ambiguity are discussed in the parser section and in the Appendix.

Sentences that include a verb are handled somewhat differently. ("IS" is not defined as a verb in the above example.) Since verb processing involves more complex data structures, a sample verb analysis is not shown until after the following description of ring structures and data input rules.

RING STRUCTURES AND DATA INPUT

In the preceding example, the fact that the organizational entity called the 638TH is a BATTALION was shown by a connective ring intersecting
the referent rings of 638TH and BATTALION. This connective ring (Fig. 2) was established as a data entry in response to the sentence,

DATA: THE 638TH IS A BATTALION!

Such data input sentences are parsed in the same way that query sentences are, using the same grammar. The difference comes only at the final step, where data input statements build connective rings in the data base.* All such rules for data input begin with the word “DATA:” and end with an exclamation mark, as in Rule 901, DATA: + R + / + R + ! → S.

The slash (/) is an arbitrary function word symbol originally used in a group of formatted data input statements, such as:

DATA: 638TH / BATTALION!

These formatted statements were used to build an initial “debug data base” and are still part of the grammar. The format was relaxed slightly by adding the rule IS + A → /. Applying this rule and Rule 11, THE + R → R, to the statement shown in Fig. 8 sets up the required inputs for Rule 901, which is then executed to form the two-link connective ring.

Three-link rings are formed similarly. Samples of various ways of inputting the fact that PARKER is COMMANDER of the 638TH are shown in Fig. 9. For any of the statements to be successful, it must be parsed down to an unambiguous characterization in a form such as “DATA: COMMANDER OF 638TH IS PARKER!” If any ambiguity remains, the data entry is rejected. As illustrated, the connective ring is directed but not ordered; i.e., no “starting point” is marked. However, when such a statement is accepted, the R-word that is used attributively (COMMANDER, in this case) is placed on a special data ring associated with the preposition OF. The use of this special ring, which in effect orders connective rings such as that in Fig. 9, is discussed in the Appendix.

A variation of the three-link connective ring is introduced to store time-dependent data. For example, if the 638TH BATTALION moved from

Figure 8. Example of the parsing of a data input sentence and the resulting formation of a two-link connective ring in the data base.

DATA: 638TH / COMMANDER / PARKER:
DATA: THE COMMANDER OF THE 638TH BATTALION IS PARKER:
DATA: LT. COL. PARKER IS THE 638TH’S COMMANDER:
DATA: LT. COL. JONATHAN M. PARKER IS COMMANDER OF THE ENGINEER BATTALION AT FT. LEWIS:

Figure 9. Examples of data input statements that result in the formation of the three-link connective ring shown.

* Rings of the data base are stored as pages on the disc and accessed from grammar rules through a paging technique.13
DEACON: DIRECT ENGLISH ACCESS AND CONTROL

Figure 10. Use of time fan to record that the 638th Battalion was at Fort Irwin between times 600 and 18000 (points on a “time line” corresponding to particular dates) and has been at Fort Lewis since 20000.

FORT IRWIN to FORT LEWIS, this could be recorded by simply changing the connective ring. However, if a locational history is desired, a different approach is required. The technique used for recording time-dependent data involves a “time fan,” as illustrated in Fig. 10.

The connective ring fans out through a “time line” and may then lead to several different “values” for the “attribute” LOCATION. In the current system, an attribute may have only one value at a given point in time, except for the instant of change.

A time fan entry includes not only a time point, but also an “aspect” marker to show whether the applicability of the associated value is just beginning, is continuing, or is ending. For example, in Fig. 10 the 638TH has the value FORT LEWIS for the attribute LOCATION for a time span beginning at the time 20000.

Time dependent data may be input by formatted statements such as

DATA: 638TH / LOCATION / FORT LEWIS / BEG 20000!

Alternatively, it may be input by a sentence such as

DATA: THE 638TH BATTALION ARRIVED AT FORT LEWIS AT 20000!

The parsing of a sentence such as this builds a “verb table,” as discussed in the following section. Currently, a separate input statement is required for each entry on the time fan, and a specific time (not a span) must be given.

One other type of data structure is used. Numeric values, stored as numbers rather than as rings, result in data “dead-ends.” For data such as PARKER’S SERIAL NUMBER, the number is appropriately unitless. However, the data structures used do not yet make allowance for units where they are required. Therefore, distances, weights, etc., are now stored and used as unitless numbers.

Attempts at inputting data are rejected by the system under various circumstances. Of course, any input statement is rejected if it is beyond the grammatical capability of the system. Also rejected are statements involving ambiguous data referents. For example, if there are two PARKERS in the data base, the following statement is rejected:

DATA: PARKER’S SERIAL NUMBER IS 96738332!

PARKER IS AMBIGUOUS

The user may restate the command, identifying which PARKER he means.

If the data being input is already recorded, the system so states:

DATA: LT. COL. PARKER’S SERIAL NUMBER IS 96738332!

INFO ALREADY IN DATA

Although the current system does have a few more capabilities that have not been discussed above, a good deal more work needs to be done on inputting data. Additional interactive features are required and planned. Also needed are the ability to input more than one connective ring per input statement, the ability to input data while defining a word, and an improved ability for inputting characteristics that apply to each item in a given class. Also, only rudimentary capability now exists for changing data entries or deleting them from the data base.

VERBS

Grammar rules involving verbs are somewhat different from those described so far. The difference is that these rules do not check data structures imme-
A verb is considered to specify an event, which is defined as the existence of a state or a change of state as indicated by a time-dependent three-link connective ring in the data base. (See Fig. 10.) Specifically, the verb is associated with the referent ring that is used as an attribute. Also, it concerns a particular aspect of the relationship—a beginning, continuing (existence) or ending of the relationship. For example, ARRIVING, VISITING, and DEPARTING are associated with a beginning, continuing, and ending, respectively, of a LOCATION.

The associated attribute, aspect, and tense of a verb are specified in its definition. This information, which is provided by the dictionary lookup as part of the initial verb phrase, forms part of the verb table.

A completed verb table and tree diagram are shown in Fig. 11 for the sentence:

HAS THE 638TH BATTALION ARRIVED AT FORT LEWIS SINCE 18000?

Rules 5 and 11 apply as previously shown to reduce THE 638TH BATTALION to 638TH. Rule 51, HAS + R + V → V, then applies to HAS 638TH ARRIVED. This rule checks the compatibility of the auxiliary verb with the tense of the main verb, and places 638TH on the verb table as Subject. The output phrase in the tree diagram is labeled ARRIVED', with the prime indicating that the verb table has been extended or modified. Other rules place FORT LEWIS on the verb table as Value and set the time span as 18000 to 20500 (the implicit current time).

After the verb table has been completed (at the node labeled ARRIVED''), the “hypothesis testing” rule, V + ? → S, searches the data base for a BEGINNING LOCATION of the 638TH at FORT LEWIS between times 18000 and 20500. Since Fig. 10 shows an appropriate arrival (at 20000), the response to the sentence is YES.

The verb routine is also used in a variation of its role as a hypothesis tester to fill in gaps in the “completed” verb table. For example, consider the sentence

WHEN DID THE 638TH ARRIVE AT FORT LEWIS?

The verb table would not actually have a gap, since the phrase DID . . . ARRIVE sets the time span as past. However, in checking the data base for structures corresponding to past arrivals of the 638TH at FORT LEWIS, the verb routine fills in the specific times for which arrivals are recorded. Rule 308, WHEN + V + ? → S, therefore executes the verb routine, retrieves the list of times (or time spans, i.e., “between ______ and ______,” in case of implied arrivals for which a definite time is unavailable) and prepares the list as the response to the sentence. Similarly, if the sentence did not specify the place of arrival, the verb routine would fill the gap with a list of places in which the 638TH had arrived. In more general terms, it fills in all appropriate values of the verb’s associated attribute for the sentence subject. However, no rules of grammar have yet been written to take advantage of this type of verb table gap-filling.

Reference 12 describes the use of list processing “generators” which are used in conjunction with the verb routine for dealing with quantifiers. In addition
DEACON: DIRECT ENGLISH ACCESS AND CONTROL

373
to quantifiers such as “any” and “all,” the concept of quantifiers/generators applies to “what” and “how many.”

In addition to its use in queries, the verb table is also convenient for inputting time-dependent data. Since it characterizes an event, a similar characterization in ring structures can be established if the verb table is specific enough. This use of the verb table was discussed earlier, in the section “Ring Structures and Data Input.”

More complex verb tables, which result from more complex sentences, are discussed in Reference 12. Additional grammar rules, both verb and non-verb, are also shown in Reference 12. The strategy for determining what rules are applied in what sequences is described in the following section.

THE LEVEL PARSER

The procedure used by the DEACON system to apply rules of grammar is a bottom-to-top, rewriting parser that produces, in parallel, all analyses of a sentence that the grammar allows. The unique feature of the parser is its use of level conventions to restrict redundant application of grammar rules.* This is an especially important consideration in a parser which was designed to use a context sensitive phrase structure grammar with discontinuous rules.

As indicated in the preceding examples, the parsing tree (Fig. 7) that represents an analysis of the sentence is built from the bottom up. The parser’s ultimate goal is to apply enough grammar rules to collapse the sentence to a single sentential phrase (S). Analysis proceeds in steps from the sentence to (hopefully) the S-phrase goal. At each step in the parsing, the sentence is represented within the computer as a list of phrases, which is called a subgoal. Such a list made up exclusively of initial phrases is called an initial subgoal. Every subgoal is a representation of the sentence, but only the phrases in initial subgoals correspond directly to terms of the sentence. Other subgoals are a partially parsed representation of the sentence and contain at least one intermediate phrase. This distinction between partially parsed subgoals and initial subgoals is ignored by the parsing procedure, but is useful in describing the parser.

The DEACON parser is a rewriting parser, which here means something more than a parser that employs rewriting rules of grammar. For, when a rule of grammar successfully applies to a sequence of phrases in a subgoal, the whole subgoal is rewritten.* The output phrase from the grammar rule is substituted for its constituents and the other phrases in the subgoal are copied. The new subgoal is added to a list of subgoals and is parsed along with them.

Thus, the list of subgoals represents all possible analyses of one initial subgoal that have been developed up to a certain point. All potential analyses are developed in parallel. In contrast to a parsing strategy that follows a single analysis as far as possible and then backtracks to get other analyses (if any), the Level Parser carries all analyses along together and discards unproductive ones.

The Level Idea

The level idea is based on the standard representation of an analysis of a sentence as a parsing tree (Fig. 12). If only context-free rules of grammar are used, a level corresponds to the height of a phrase within the parsing tree. Phrases in an initial subgoal

*Although it is not necessary to rewrite the whole subgoal, the context phrases in context sensitive rules and the intervening phrases in discontinuous rules must be rewritten. A new parser which rewrites only what is necessary, but which does not use levels, has been implemented.

†There may be more than one initial subgoal for a sentence if terms of the sentence have alternative syntactic definitions. In such cases, each initial subgoal is parsed separately due to lack of core space.
are assigned level zero; they are at the bottom of the parsing tree. Any phrase produced by combining phrases is represented as a node which is higher in the parsing tree than the nodes which represent its constituents. A phrase which is produced by application of a grammar rule to phrases in the initial subgoal is, for example, assigned a level of one. This intuitive idea of level was modified slightly so that context sensitive rules could be used. This modification is necessarily detailed and is described in conjunction with the two simplified parsing examples below.

Using the level idea, the parser first combines phrases at the bottom of the parsing tree. When all that can be done at this lowest level has been done, the initial subgoal is discarded and the parser considers the next higher level, or level one. When all subgoals that are at this level have been used, they are discarded and the next higher level is considered. Finally, when the subgoals at some level are discarded, there is nothing left to parse.

**The Level Conventions**

The level conventions and the general path of the parser work in conjunction to reduce redundant application of grammar rules to the same sequence of phrases. The level conventions are given below; the path of the parser is described by fiat in the two examples which follow the level convention description.

The parsing routine keeps a *master level* which defines the level within the parsing tree where the parser is currently working. Because the parsing routine begins at the bottom of the parsing tree, the master level is initially zero.

Each phrase in every subgoal is assigned a level. The level for initial phrases is zero. Any phrase output by a grammar rule is assigned a level equal to one plus the master level. Unaffected phrases in the subgoal are copied with whatever level they happen to have.

As the parser systematically matches phrases in the subgoals to rules, the following level restrictions must be met:

1. No rule of grammar will be applied if any of its input phrases has a level greater than the current master level.
2. No rule of grammar will be applied to a sequence of phrases unless at least one phrase in the sequence has a level equal to the master level.

Restriction 1 postpones the processing of recently produced phrases until the parsing routine begins work at the next higher level. Restriction 2 insures that the phrases at a lower level, which have already been processed, will not be done again.

**Examples**

Table 1 shows an abstract example where the level conventions are used to reduce redundant parsing. (Unfortunately, an interesting example is too lengthy for presentation at this time.) The example also gives a more detailed explanation of the order of parsing.

<table>
<thead>
<tr>
<th>Master level = 0</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial subgoal E</td>
<td>1) A + B — E</td>
</tr>
<tr>
<td>2) C — F</td>
<td></td>
</tr>
<tr>
<td>3) E + F — G</td>
<td></td>
</tr>
<tr>
<td>After application of Rule 1 in the initial subgoal</td>
<td></td>
</tr>
<tr>
<td>Initial subgoal E</td>
<td></td>
</tr>
<tr>
<td>Subgoal 1 F</td>
<td></td>
</tr>
<tr>
<td>Subgoal 2 G</td>
<td></td>
</tr>
<tr>
<td>After application of Rule 2 in the initial subgoal</td>
<td></td>
</tr>
<tr>
<td>Initial subgoal E</td>
<td></td>
</tr>
<tr>
<td>Subgoal 1 F</td>
<td></td>
</tr>
<tr>
<td>Subgoal 2 G</td>
<td></td>
</tr>
<tr>
<td>Subgoal 3 H</td>
<td></td>
</tr>
<tr>
<td>After application of Rule 3 in subgoal 2</td>
<td></td>
</tr>
<tr>
<td>Subgoal 1 I</td>
<td></td>
</tr>
<tr>
<td>Subgoal 2 J</td>
<td></td>
</tr>
<tr>
<td>Subgoal 3 K</td>
<td></td>
</tr>
<tr>
<td>Subgoal 4 L</td>
<td></td>
</tr>
</tbody>
</table>

### Table I. Subgoals produced by the Level Parser in a simplified context-free example.

1. No rule of grammar will be applied if any of its input phrases has a level greater than the current master level.
2. No rule of grammar will be applied to a sequence of phrases unless at least one phrase in the sequence has a level equal to the master level.
The initial subgoal is made up of three phrases with categories A, B, and C, respectively. Each initial phrase is given level zero (indicated by subscripts). The master level is initially zero.

Parsing begins at the left on phrase A in the initial subgoal. Rule 1 matches phrases A and B and after successful application, the initial subgoal is rewritten incorporating the new phrase E, which is assigned a level one.

The parser moves down the first column to the phrase E in subgoal 1. Consideration of this phrase is aborted because the level of E is greater than the master level. Parsing of phrase E is postponed until the master level is raised to one.

The parser moves to the top of the next right column and finds that no rules apply to phrase B. Moving down the second column, Rule 2 applies to phrase C in subgoal 1. New subgoal 2 is added to the bottom of the subgoal list. Phrase F in subgoal 2 is out of range and Rule 2 applies to phrase C in the initial subgoal to produce subgoal 3. Continuing, the parser finds that there are no 3rd phrases in subgoals 1 and 2 and that phrase F in subgoal 3 is out of range.

The parser has completed its first left-to-right sweep. Everything that can be done strictly at the bottom of the parsing tree has been done. The master level is raised to one. Now each sequence of phrases to which a rule can apply must have at least one phrase at level one. Because no sequence of phrases in the initial subgoal can meet this requirement, the initial subgoal is discarded.

The general problem in allowing context sensitive rules is that of having the context available when it is needed. Because the DEACON parsing method rewrites entire subgoals, the context is always available when needed. The problem, instead, becomes how to insure that the context is properly parsed after it has been used as context. Consider the example in Table II.

Following the parsing procedure outlined in the previous example, Rule 4 and Rule 5 are applied to the initial subgoal V W X, resulting respectively in subgoals 1 and 2. At the end of the first left-to-right sweep, there are three subgoals, namely the initial subgoal, subgoal 1 and subgoal 2. The master level is raised to one and the initial subgoal is discarded. Subgoal 1 is obviously a blind alley; there is no way to parse it to completion because in creating the Y-phrase, the context W which is required to parse the X-phrase was used up.

Subgoal 2, on the other hand, represents the case where the context W has been properly used to parse the X-phrase into a Z-phrase. But now what can be done with the phrase sequence V W? If the intuitive level idea is followed strictly, it is obvious that phrases V and W are still at the bottom of the parsing tree and hence each might have a level zero. But because the master level has been raised to one, level restriction 2 will prevent the re-application of Rule 4 to the V W sequence. In this case, no further rules could be applied to subgoals 1 and 2. When the master level is raised to 2, all subgoals that do not contain at least one phrase with level 2 are discarded. Subgoals 1 and 2 are discarded and there

Table II. What level should be assigned a phrase which has been used as context?

<table>
<thead>
<tr>
<th>Master level = 0</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Subgoal:</td>
<td>V_0 W_0 X_0</td>
</tr>
<tr>
<td></td>
<td>4) V + W \rightarrow Y</td>
</tr>
<tr>
<td></td>
<td>5) W + X \rightarrow W + Z</td>
</tr>
<tr>
<td></td>
<td>6) Y + Z \rightarrow S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>After application of Rule 4 in the initial subgoal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Subgoal V_0 W_0 X_0</td>
</tr>
<tr>
<td>Subgoal 1</td>
</tr>
<tr>
<td>Y_1 X_0</td>
</tr>
<tr>
<td>After application of Rule 5 in the initial subgoal</td>
</tr>
<tr>
<td>Initial Subgoal V_0 W_0 X_0</td>
</tr>
<tr>
<td>Subgoal 1</td>
</tr>
<tr>
<td>Y_1 X_0</td>
</tr>
<tr>
<td>Subgoal 2</td>
</tr>
<tr>
<td>V_0 W_1 Z_1</td>
</tr>
</tbody>
</table>

In actual practice, phrases with category S are not rewritten into the subgoal list, but are set aside on a special output list.
Table III Parsing continued after appropriate assignment of level to context phrase.

<table>
<thead>
<tr>
<th>Master level = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>After application of Rule 4 to subgoal 2</td>
</tr>
<tr>
<td>Subgoal 1: Y₁ X₀</td>
</tr>
<tr>
<td>Subgoal 2: V₀ W₁ Z₁</td>
</tr>
<tr>
<td>Subgoal 3: Y₂ Z₁</td>
</tr>
</tbody>
</table>

is nothing left to parse. No analysis of the sentence V W X could be found.

To correct this undesirable state of affairs, the level idea is pragmatically extended. Any phrase “output” by a grammar rule is assigned a level equal to one plus the master level. Phrases “output” from a grammar rule are the single new phrase and any phrases which should be retained as signalled by the grammar rule. Assignment of level to the new phrase corresponds with the intuitive level idea. The phrases which are to be retained are the context phrases. So any phrase used as context is also assigned the new level and not just rewritten with whatever level it happened to have.

A review of Table II with this new level convention in mind will show that the W phrase in subgoal 2 is assigned level 1. Then the sequence V W meets the requirement of having at least one phrase with level equal to the master level and Rule 4 may be applied. (The phrase history list is consulted to obtain this previous result directly.) The problem of what to do with context phrases is resolved, as shown in Table III.

At the end of the left-to-right sweep, subgoals 1 and 2 are discarded as above, but this time there is a subgoal left to parse. Application of Rule 6 to subgoal 3 results in the desired completion of parsing.

This slight modification to the level conventions changes the nature of the idea of level. Instead of “height within the parsing tree,” the levels, therefore, no longer say something about the phrase to which they are attached. Instead, they indicate something about the subgoal containing the phrase to which they are attached.

HARDWARE CONFIGURATION

The current experimental DEACON system is implemented on a 16K GE-225 general purpose digital computer with a 20-bit word length and an 18 microsecond cycle time. A random access disc unit of six million words is used for secondary storage. Model 33 or Model 35 teletypes connect to the mainframe through a DATANET-15 interface unit, which allows interactive use but no time-sharing. The hardware configuration also includes a card reader, a punch, a printer, and six tape units. These devices are used for service jobs such as building the system, assembling programs, and taking dumps.

CONCLUSIONS

The DEACON system is in its second phase of development as a management information system featuring English language data input and query capabilities. The first phase, the DEACON Breadboard, proved the feasibility of using interpretation rules for relating English statements to computer-stored data structures. The data base was pre-stored in list structure form as a static part of the system. No time-dependent (i.e., historical or projected) data was permitted. Queries could be given in English, on punched cards, and system responses were by printer. Peripheral memory was magnetic tape, and a considerable part of the development effort consisted of devising techniques for using list processing in conjunction with peripheral memory.

The present system includes the following major advances over the Breadboard system: (1) the provision of direct access to the computer via teletype, permitting interactive use for developing the system and experimenting with it; (2) the ability to input data from the teletype in English; (3) the use of ring-type data structures, which have proven richer than the earlier list structures; (4) the use of disc peripheral memory rather than tape; (5) the use of generally more efficient processing routines such as the parser and the dictionary and data handling techniques; (6) the incorporation of time-dependent data; and (7) the use of a more generalized method for handling verbs.

Future work will concentrate on adding new features as well as extending and improving the efficiency of current capabilities. Perhaps the most significant new feature needed is the ability to define vocabulary terms in English, using previously defined terms. The importance of this capability is discussed in reference 9 as “recursive definition and the recursive behavior of structural modification.” Some initial work is being done on this and on algebraic grammar rules. Other major areas of planned grammar
extension include pronouns (with some cross-sentence referencing), negation, and clausal modification. The grammar currently consists of some 250 rules. Various techniques are under consideration to make it easier to write grammar rules.

Data structures are receiving much attention as a promising area for increasing the semantic capability of the system, and for improving processing efficiency. The processing systems such as the parser and dictionary and data handling techniques are also being improved apart from the data structures work.

Sentence processing time varies normally from under a minute to three or four minutes, but has gone as high as 17 minutes. System improvements as mentioned above, greater use of machine-language programming, and use of a larger, faster machine should much more than offset the slowing tendencies to be expected from enlarging the grammar and data base.

Our experience with the DEACON system has convinced us of the soundness of its theoretical basis. Conclusive proof may await the test of a prototype operating in a live, rapidly changing context. However, we look forward to such an operation as a source of guidance for continuing development of the DEACON system.

APPENDIX: SAMPLE RULES AND SENTENCES

Some R-phrase modification rules have been illustrated, such as Rule 5, \( R_i + R_2 \rightarrow R_1 \), and Rule 8, \( R_i + OF + R_2 \rightarrow R_2 \). There are several other rules that take two R-phrases or two R-phrases and a preposition as input, and produce an R-phrase as output. For example, Rule 1 is \( R_i + R_2 \rightarrow R_2 \) in which the output R-phrase has the same data referent as the second input R-phrase. It is needed for strings like

\[
\begin{align*}
R_1 & \rightarrow R_2 \\
\text{LT. COL.} & \rightarrow \text{PARKER}
\end{align*}
\]

This rule checks for existence of an appropriate connective ring between the referent rings of LT. COL. and PARKER. If satisfied, it uses the referent ring of PARKER as the referent ring of the output phrase. That is, it verifies that PARKER is a LT. COL. before accepting the string LT. COL. PARKER as a syntactic unit. Or, assuming that there is more than one PARKER but only one is a LT. COL., this rule selects the appropriate referent ring for the output phrase. As shown in Figure 13, the name PARKER is ambiguous in the input phrase because it leads to three distinct referent rings.

In this case, the data check resolved the ambiguity. Obviously, if the data base had been different, some or all of the ambiguity might have remained; or, if no PARKER were a LT. COL., the output R-phrase would be marked VACUOUS DESCRIPTION. If no successful analysis results from parsing LT. COL. and PARKER differently (conceivably in something like “WHO WAS THE LT. COL. [THAT] PARKER SUCCEEDED?”), the VACUOUS DESCRIPTION (LT. COL. PARKER) is typed out as a clue to the failure to answer the query.

Since Rule 1 and Rule 5 each require two adjacent R-phrases as constituents, an attempt is made to apply both rules to LT. COL. PARKER (in alternative parsings). If successful, Rule 1 interprets the phrase as PARKER, and Rule 5 interprets it as LT. COL. However, the connective ring shown in Figure 13 does not satisfy the requirements of Rule 5, so this possible ambiguity is quickly resolved. Both rules do successfully apply to the string 638TH BATTALION as shown in Figure 14. But since the referent ring of BATTALION does not link to that of COMMANDER, the parsing in which Rule 1 was used is aborted at the next step. If such ambiguities are not resolved, and result in different answers to a query, the alternative answers are typed out along with their respective parsings.

Rule 2, \( R_1 + R_2 \rightarrow R_3 \), introduces a new type of output phrase referent ring—a “scratch” ring created by the rule and linked by connective ring to appro-
Figure 14. Ambiguous interpretation (by two alternative rules) of a given phrase resolved by a data check at a higher level phrase.

appropriate referent rings in the permanent data base. The application of this rule to the string ENGINEER BATTALIONS causes such a scratch ring to be formed. (Fig. 15.) This rule also illustrates an interesting aspect of ambiguity. Although each constituent R-phrase is unambiguous, the combination may be ambiguous because their data referents are interconnected via alternative paths. For example, if some BATTALIONS are HEADQUARTERED at FORT IRWIN and others are temporarily LOCATED there but HEADQUARTERED elsewhere, an ambiguous response would result from the statement

LIST FORT IRWIN BATTALIONS,
BATTALION (HEADQUARTERS) FORT IRWIN
522ND
436TH
593RD
BATTALION (LOCATION) FORT IRWIN
638TH
94TH
523RD
117TH

Two separate scratch rings are created in this example—one for each interpretation.

There are also prepositional forms of these rules:

Rule 8: \( R_1 + OF + R_2 \rightarrow R_3 \)
Example: COMMANDER OF 638TH \( \rightarrow \) PARKER

Rule 31: \( R_1 + OF + R_2 \rightarrow R_3 \)
Example: PARKER OF 638TH \( \rightarrow \) PARKER

Rule 35: \( R_1 + OF + R_2 \rightarrow R_3 \)
Example: BATTALIONS OF FORT IRWIN \( \rightarrow \) FORT IRWIN BATTALIONS
(R_3 refers to two scratch rings as in earlier example.)

Rule 36: \( R_1 + IN + R_2 \rightarrow R_3 \)
Example: BATTALIONS IN FORT IRWIN \( \rightarrow \) FORT IRWIN BATTALIONS
(“IN” eliminates the location/headquarters ambiguity.)

The dual application of Rule 8 and Rule 31 to “R OF R” expressions produces ambiguity that tends to remain unresolved in such questions as WHO IS COMMANDER OF 638TH? With both rules applying, the response would be

(305 WHO IS (8 COMMANDER OF 638TH) ?)

PARKER

Figure 15. Creation of a scratch block, linked by connective rings to referent rings in the permanent data base, during application of a grammar rule in analyzing a query.
Actually, the restrictiveness of the pronoun "WHO" eliminates the second output in this example unless the system has been told that the word COMMANDER itself is in the range of the pronoun WHO.

In practice, a more general ambiguity resolver eliminates Rule 31’s application sooner. Under “Ring Structures and Data Input,” it was stated that input statements of the form “R1 OF R2 IS R3” caused the data referent of R1 to be placed on a special OF referent ring as a reminder that it may be used attributively. Rules 8 and 31 compare their R1 data referents with the OF ring. Rule 8 applies if R1 is on the OF ring, and fails if it is not; Rule 31 applies if R1 is not on the OF ring, and fails if it is. These complementary checks, which resolve the PARKER/COMMANDER type ambiguity, amount to adding the part of speech “ATTRIBUTE”.

These few rules have been presented in some detail to give the “flavor” of the problems involved in developing the DEACON grammar. These and similar rules used recursively and in combination with the verb rules, data input rules, and sentence-forming rules such as

Rule 301: LIST + R + : → S
Rule 305: WHAT IS + R + ? → S
Rule 307: WHERE IS + R → S,

successfully analyze a wide range of sentences involving attributive relationships. Some sample sentences that the system now accepts are shown with their responses in Fig. 16.

1. List all engineer battalions.
   425 TH
   638 TH
   94 TH
   523RD
   682 ND
   496 TH
   117 TH

2. List the commanders of engineer battalions.
   J.C. Irvin
   D.J. James
   J.M. Parker
   S.V. Olsen
   T.C. Allen
   I.S. Parker
   J.C. Conway

3. Where is Parker?
   Ft. Lewis
   [Ambiguous = 2 Parkers]

4. Where is Lt. Col. Parker?
   Ft. Lewis
   [Only one is a Lt. Col.]

5. When will Lt. Col. Parker leave Ft. Lewis?
   22000
   [Times are currently treated as relative numbers on a "time line."]

6. It is now 23000!
   OK
   [There is now no system clock yet. The user must set "now" explicitly or it is automatically "0,0"]

7. Where is Lt. Col. Parker now?
   Presidio

8. What is the date of rank of the commander of the 638 TH Battalion of Fort Lewis?
   11600

9. What is Olsen’s date of rank?
   12900

10. Does Parker’s date of rank precede Olsen’s date of rank?
    Yes

11. What staff colleges has Lt. Col. Parker attended?
    Command General School
    Infantry School

12. Has Lt. Col. Parker arrived at the Presidio from Ft. Lewis?
    Yes

13. What is the readiness of Lt. Col. Parker’s Battalion?
    Orange

14. Data: The 425 TH will leave Ft. Lewis at 21950!
    OK

15. Data: The 638 TH will arrive at Ft. Lewis at 20000!
    OK

16. Is the 638 TH scheduled to arrive at Ft. Lewis before the
    425 TH leaves Ft. Lewis?
    Yes

17. How many combat engineer companies have trained at
    Ft. Irwin this year?
    6

18. How many Sixth Army installations are there?
    38

19. What is the distance from the 638 TH Battalion to San Diego?
    61
    [No units are currently stored. Forts and cities are located on a square world of xy coordinates, so "distance" is the "square root of the sum of the squares..."]

Figure 16. Sample sentences.
ACKNOWLEDGMENTS

Our greatest debt is to Dr. Frederick B. Thompson, who is clearly the author of DEACON, if not of this paper. Mr. John W. Gwynn and Mr. Gregory D. Gibbons contributed to the Breadboard and the experimental system. Mrs. Jacque Suzanne Pruett was a valuable member of the Breadboard team, and Mr. Russell J. Abbott, Mr. Robert L. Price, and Mrs. Marillyn Popkin have contributed to the experimental system since its beginning. Dr. John C. Fisher adeptly pulled together the enabling resources for the experimental system. The project is under the management of Dr. Herbert R. J. Grosch.

REFERENCES