Inform. Stor. Retr. Vol. 4, pp. 13–26. Pergamon Press 1968. Printed in Great Britain

RETRIEVAL FORMULAE FOR INQUIRY SYSTEMS

JACQUES F. VALLEE, GILBERT K. KRULEE and ALBERT A. GRAU

The Technological Institute, Northwestern University

Summary—This article discusses the design of automated information-retrieval systems which accept questions in ordinary English and yield as output a numerical answer. Fundamehtal to the efficient implementation of such a system is the notion of retrieval formulae. These formulae are representations of search strategies in the artificial language of a retrieval automaton. As a first step, a linguistic processor maps English questions into this artificial language. Subsequently, the system will conduct the appropriate search to satisfy the requirements of each formula. The method is illustrated with references to a FORTRAN program which retrieves information from a data base obtained from the field of Astronomy.

I. INTRODUCTION

THE NEED for immediate interrogation of stored information has been a critical one in many areas of industrial management and scientific research. Typically, one may want to know the number of objects that possess certain properties within a given population or to compare different populations in terms of the properties possessed by their members. In many cases, the populations under study are large and the problem of designing high-speed and efficient search strategies is therefore of considerable importance.

A second, and theoretically more interesting, problem is that of the language used in formulating the questions of information retrieval, i.e. *the vehicle of the assignment of a task to the searching automaton.* Given any physical object R and n characteristics or attributes which describe this object (and which may be of a qualitative or quantitative nature), we assume that the properties of R have been converted by an appropriate coding procedure into the form of a word of information in storage:

$$\sigma(R) = v_1 v_2 \ldots v_n. \qquad (1)$$

This word of information will be called the "signature" of R. Its elements are alphanumerical symbols. The set of physical objects which is susceptible to quantification through such a formal coding scheme constitutes the data base of the question-answering automaton. All problems presented to the system will be interpreted in terms of this same code: thus, given a set of coded entities, S, and a verbal statement of certain properties (P) expressed in the same code, we shall call a *Search Strategy* any finite algorithm leading uniquely to the partition:

$$S = S_1 \cup S_2,$$

where S_1 is the set of elements of S which have the properties (P).

The fact that such an algorithm, if it exists, is finite, does not require a formal proof since for all practical purposes the set S is finite and each element of S has a finite number of attributes. Given (P) it can not be guaranteed that a search strategy exists which will result in the partition of S into S_1 and its complement. In particular, if (P) contains contradictions, no search strategy can be found because S_1 is empty. If the statement is ambiguous,

S_1 may not be uniquely defined. Therefore the analysis of search strategies leads immediately to questions concerning the relevance and possible ambiguity of the input statements. It should be realized here that responsibility for assigning meaningful tasks to the system rests to a large extent with the user.

II. RETRIEVAL FORMULAE

Elementary English questions regarding the characteristics of the data base may be of two types. Either one asks for a *number*, namely, the cardinality of a certain subset of the given population, or for a *proportion* of objects having certain characteristics among a certain sub-population whose cardinality may or may not have been previously determined. We recognize the importance of this difference between the two types of questions by calling them "questions of type n" and "questions of type p", respectively.

A *semantically valid* question is one which specifies without ambiguity the properties of all subsets to be extracted: it must provide the processor with adequate information on the relevant set of signatures. The machine representation of this scheme is greatly clarified if, by convention, we denote as

$$Y = (v_1 v_2 \ldots v_n) \tag{2}$$

the cardinality of the subset (V) of the data-base defined by selecting all objects whose signature is: $v_1 v_2 \ldots v_n$. Given a second subset (W) whose elements all have the signature: $w_1 w_2 \ldots w_n$, it is natural to denote the cardinality of their union as:

$$Y = (v_1 v_2 \ldots v_n) + (w_1 w_2 \ldots w_n) \tag{3}$$

since the two subsets are disjoint. More generally, the number:

$$Y = (u_1 u_2 \ldots u_n)/[(v_1 v_2 \ldots v_n) + (w_1 w_2 \ldots w_n)] \tag{4}$$

represent by convention the proportion of elements of $V \cup W$ which have signatures of the form: $u_1 u_2 \ldots u_n$. Thus it corresponds to a question of type p while the expressions seen earlier were associated with type n.

In practice, it is often observed that all questions do not involve every characteristic of the objects, and the corresponding digits of the signature are left undefined. Assuming for instance that the first attribute is to be ignored, we would replace it by a dot and rewrite expression (2) as:

$$Y = (.v_2 \ldots v_n). \tag{5}$$

Any expression of the forms (2)–(5) will be called a *Retrieval Formula.*

Noting that the value of each term in a retrieval formula, i.e. the cardinality of the subset it defines, can be obtained by a call to a subprogram which performs a conventional search, we will now describe a procedure that compiles a program corresponding to any well-formed formula—leading to the appropriate numerical answer.

For clarity it will be useful to formalize the rules for generation of valid retrieval formulae:

$S \rightarrow T = \alpha$	Type n formula
$\rightarrow Y = \alpha/\alpha$	Type p formula
$\alpha \rightarrow (v_1 v_2 \ldots v_n)$	Terminal signature
$\rightarrow (y+y)$	
$y \rightarrow (v_1 v_2 \ldots v_n)$	Terminal signature
$\rightarrow y+y$	Recursion rule.

A finite-state grammar for the generation and processing of retrieval formulae can clearly be derived from these rules.

The set of strings generated by this procedure is the language of the information retrieval system. To each formula will be attached two indicators $N1$ and $N2$. They can be thought of as the number of terms in the numerator and the denominator of the retrieval formula. In particular, $N2 = 0$ characterizes type n questions and $N2 > 0$ characterizes type p questions.

Let us define the following operations:

$\phi 1$: Set $k = N1 = N2 = 0$
$\phi 2$: Set $k = 1$
$\phi 3$: Read n symbols into pushdown-stores $P1$ to Pn, increment $N1$ by 1
$\phi 4$: Read n symbols into pushdown-stores $P1$ to Pn, increment $N2$ by 1
$\phi 5$: Set $k = 0$
$\phi 6$: Backspace.

Also consider the logical propositions:

$K0$: "k is equal to zero"
$K1$: "k is equal to 1"
$I1$: "$N1$ is greater than 1"
$I2$: "$N2$ is greater than 1".

Using these operations and propositions, we can write a simplified state-diagram for a recognizer which accepts retrieval formulae as inputs. This recognizer (Fig. 1) performs one important function similar to that of a compiler. It loads the parameters of each elementary formula into n-pushdown-stores and provides the key to the operations to be performed upon them. Execution of the sequence of operations thus set up in table form is then straightforward.

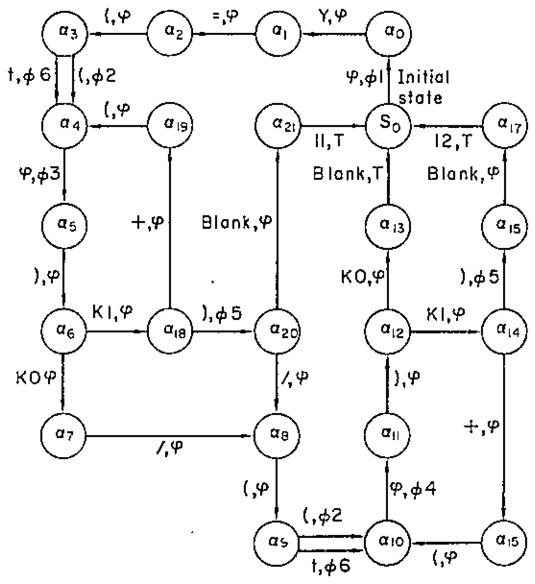

FIG. 1. An acceptor for retrieval formulae. Each transition is represented as an arrow between two states. The pair (input, output) is given for each transition. The symbol φ denotes the no-operation code. t means "all other input symbols". T represents the acceptance of the formula.

III. A LANGUAGE PROCESSOR

Fig. 2 shows the organization of a language processor that we wish to describe. Let Λ be the input set of English statements. Subprogram A is the language processor itself, i.e. an automaton capable of recognizing Λ words, while subprogram B describes their meaning by reference to a thesaurus of technical descriptors and their corresponding subroutines. The solution to the given English question, as formulated by A and B, is given by automaton C, which is driven by the retrieval formulae generated by B.

The operation of C has been discussed in detail in the first part of this article and follows essentially the program outlined in Fig. 1. It remains to show how our program maps English statements into the artificial language of retrieval formulae. Before proceeding, we need to describe briefly the context of this particular implementation and the signatures

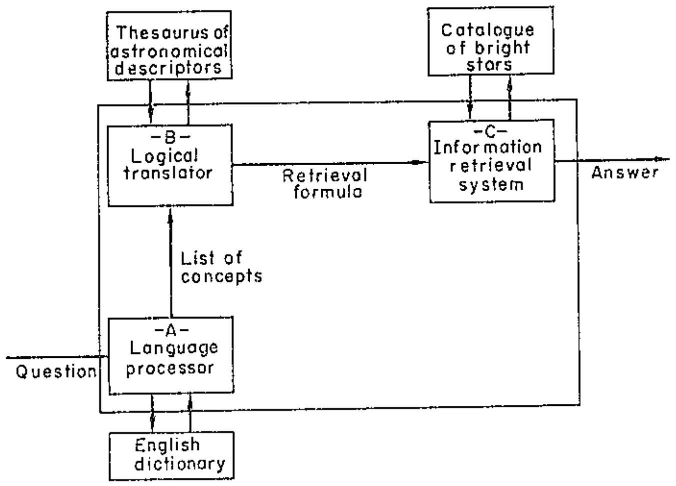

FIG. 2. Organization of ALTAIR.

which are possible under the current system, which is called ALTAIR (Automatic Logical Translation and Information Retrieval). Each signature contains four symbols which represent four essential characteristics of a stellar system: (i) the position of the star on the color-luminosity diagram, (ii) its spectral type, (iii) its multiplicity configuration and (iv) its total space velocity. A discussion of this coding scheme, with algorithms for converting the characteristics of stellar systems into codes has been presented elsewhere[1]. Given this coding scheme, the first section of the program, A, locates key-words in the input sentence. The second section attaches significance to these particular words and produces as output a suitable set of signatures, which constitute a formal problem representation.

The organization of subprogram A depends upon our assumption that, in this implementation, all questions will be of type n or type p and that questions of these two types can only be expressed using a technical vocabulary which is flexible but not unlimited. Therefore, there are a number of possible types, or patterns, of input sentences in the *formal language* from which the retrieval formulae can be derived. These patterns can be described completely by use of the following definitions and notations:

Let *N* by any instruction of type *n* which maps into the concept "COMPUTE-A-NUMBER".
P an instruction of type *p* which maps into "COMPUTE-A-FRACTION".
C the context, i.e. the environment of the system. In this implementation,
C is either the empty string or a string which maps into the concept "IN-THE-CATALOGUE".
D a delimiter which maps into the concept "AMONG".
B any expression which maps into the concept "SUCH THAT".

These are the *formal constituents* of the input string. To every formal constituent is now attached an alphabetic symbol.

Following the Backus convention, we shall enclose the *generic symbols* of the string in brackets. We can now express the fact that the highest level string is a question that may be of type *n* or type *p*, and which involves certain technical *definitions*. These definitions are composed of technical terms that may contain simply the statement that a certain characteristic of a star belongs to a certain range, or more complex statements specifying the properties of two, three, or even all four characteristics available under our system of codes. This formal language can be described as follows:

```
⟨question⟩::=⟨type n string⟩|⟨type p string⟩
⟨type n string⟩::=N⟨definition⟩C
⟨type p string⟩::=N⟨definition⟩D⟨definition⟩C|
                  P⟨definition⟩D⟨definition⟩C|
                  P⟨definition⟩C
⟨definition⟩::=⟨entity⟩|⟨entity⟩⟨specification⟩
⟨specification⟩::=B⟨characterization⟩|
                  B⟨characterization⟩⟨specification⟩
⟨characterization⟩::=⟨attribute 1⟩⟨range 1⟩|
                     ⟨attribute 2⟩⟨range 2⟩|
                     ⟨attribute 3⟩⟨range 3⟩|
                     ⟨attribute 4⟩⟨range 4⟩|
⟨attribute 1⟩::=HR REGION
⟨attribute 2⟩::=SPECTRUM
⟨attribute 3⟩::=MULTIPLICITY
⟨attribute 4⟩::=VELOCITY
⟨range 1⟩::=DWARF|GIANT|SUBGIANT|SUPERGIANT|UNKNOWN
⟨range 2⟩::=BELOW ⟨spectral type⟩|ABOVE ⟨spectral type⟩|
            BETWEEN ⟨spectral type⟩ AND ⟨spectral type⟩
⟨range 3⟩::=SINGLE|MULTIPLE|TRIPLE|BINARY|SPEC. BINARY|VISUAL-BINARY|CPM
⟨range 4⟩::=BELOW ⟨speed⟩|ABOVE ⟨speed⟩|BETWEEN ⟨speed⟩ AND ⟨speed⟩
⟨speed⟩::=⟨number⟩|⟨number⟩ KM/S
⟨entity⟩::=STAR|PRIMARY
```

What is meant by "number" and by "spectral type" is for the implementer to decide. In ALTAIR we have taken:

```
⟨spectral type⟩::=B2|B3|B7|B8|A2|A3|A7|A8
                  F2|F3|F7|F8|G2|G3|G7|G8
                  K2|K3|K7|K8|M2|M3
⟨number⟩::=10|20|30|40|50|60|70|80
```

The following examples will help illustrate this system of rules.

First example

⟨question⟩::=⟨type n string⟩
⟨type n string⟩::=N ⟨definition⟩ C
⟨definition⟩::=⟨entity⟩
⟨entity⟩::=STAR
The input string would be: "What is the number of stars".
(Here the context, C, is the empty string.)

Second example

⟨question⟩::=⟨type n string⟩
⟨type n string⟩::=N ⟨definition⟩ C
⟨definition⟩::=⟨entity⟩⟨specification⟩
⟨entity⟩::=STAR
⟨specification⟩::=B ⟨characterization⟩⟨specification⟩
⟨characterization⟩::=⟨attribute 4⟩⟨range 4⟩
⟨specification⟩::=B ⟨characterization⟩
⟨characterization⟩::=⟨attribute 2⟩⟨range 2⟩
The following input string would fit this formal representation:
"Find how many stellar systems have a speed below fifty kilometers per second and a spectrum bluer than $G8$."

Third example

⟨question⟩::=⟨type p string⟩
⟨type p string⟩::=P ⟨definition⟩ C
⟨definition⟩::=⟨entity⟩⟨specification⟩
⟨specification⟩::=B ⟨characterization⟩
⟨characterization⟩::=⟨attribute 2⟩⟨range 2⟩

With the input string: "Compute the proportion of stars whose spectrum is between $F3$ and $F7$."

Note: The system will understand that the *percentage* of such stars *in the whole catalogue* is requested.

It is interesting to remark that this system of rules, although it gives the user the advantage of a very flexible formulation of his questions, defines quite strongly the input pattern. For example, varying formulations of an instruction like "how many", "find the number of", "what is the cardinality of the set of", "find how many", "calculate how many", etc., will all be mapped into a single descriptor: "COMPUTE-A-NUMBER".

Only six types of semantic units will be distinguished by this reduction:

1. Instructions such as "COMPUTE-A-NUMBER", "COMPUTE-A-FRACTION".
2. Technical terms such as "STAR", "STELLAR SYSTEM".
3. Attributes of objects such as "SPECTRUM", "VELOCITY".
4. Terms which specify a range of values: "BLUER THAN $K2$".
5. Keywords such as "AMONG" and "SUCH THAT".
6. Context-defining terms: "IN THE CATALOGUE".

Investigation of the expected concept lists leads us to the problem of extracting the above elements from their varying formulations, of going from the "informal" input language to the formal patterns we have just defined. The objective of this analysis is to permit the reduction of a list of dictionary entries into a smaller list whose elements are indexes of general concepts instead of references to individual words. In our Fortran-based implementation, we have obtained this reduction through a system of lists using indirect addressing. We will offer comments on a possible parallel between this structure and that of IPL-V lists. The identification of the concepts proceeds according to the following algorithm:

Let w_n be the n-th word in the input sentence (after elimination of non-substantive terms such as "THE"). Assume it has been identified as dictionary entry i. Let B^4 and B^5 be two lists of system-supplied constants.

To i, there corresponds a pair of elements (B_i^4, B_i^5). Let us first replace w_n by B_i^5. Then we consider B_i^4:

(A) If $B_i^4 = 0$, this means by definition that word w_n does not appear as the initial term in a compound-meaning unit. Therefore we go to the next word and continue to apply the algorithm.

(B) If $B_i^4 \neq 0$, this indicates that a compound-meaning unit has been encountered.

In this case, B^5 is an indirect-addressing device which works as follows: Set $k = B_i^5$. Compare B_k^4 with w_{n+1}. If they are equal, reorder the input string in the following manner:

$$\begin{aligned}[w_j] &= w_j \quad && \text{for } j \leq n \\ &= w_{j+1} \quad && \text{for } j > n.\end{aligned}$$

Then, set $k = B_k^5$ and replace w_n by k. If $k \leq 150$, identification is complete, go to w_{n+1}. If not, continue the comparison using w_{n+1}.

If $B_k^4 \neq w_{n+1}$, the word which we expected to complete or to continue the compound-meaning unit has not been found. We perform the same comparison again using the next element in the list B^4 (namely, B_{k+1}^4) provided it refers to the same entry. Continue this process until equality is finally reached, or until an element of B^4 is found which does not refer to the current entry. In this case, the program will replace w_n by B_i^4, usually a "trap" indicating failure of the algorithm, i.e. illegal input string.

Example: Consider the question

"HOW MANY STARS ARE INCLUDED IN THE BRIGHT STAR CATALOGUE"

Subprogram A has reduced "STARS" to its singular form "STAR" and it has eliminated the words "THE" and "ARE" which do not play an active part in the assignment of meaning to the other words. Furthermore the remaining words have been identified in the dictionary and replaced by a number. The input string is now in the form:

/126/127/033/132/137/133/033/135/.

In Table 1 is a representation of the corresponding section of the lists, where the line number (k) appears with the entry it refers to and the values of (B_k^4, B_k^5). The first word (w_1) is "HOW". It was recognized as dictionary entry 126. We find that:

$$B_{126}^4 = 077 \neq 0.$$

Therefore we set:

$$k = B_{126}^5 = 210.$$

Now, $B_{216}^4 = 127$ and also the next word $w_2 = 127$. Thus a compound-meaning unit has been encountered. We destroy w_2 and we re-order the string. Finally, we replace w_1 by

Table 1. Example of the analysis lists used in the interpretation of a question

Word	k	entry	B^4_k	B^5_k
STAR	033	033	000	017
HOW	126	126	077	210
MANY	127	127	000	077
INCLUDED	132	132	066	224
BRIGHT	133	133	077	230
CATALOGUE	135	135	000	077
IN	137	137	077	231
	210	126	127	065
	224	132	137	225
	225	225	133	227
	227	227	033	228
	228	228	134	069
	229	228	135	069

$B^5_{210} = 065$. The next word is "STAR" with $k = 33$. Since $B^4_{33} = 0$, we replace "STAR" directly by the number B^5_{33}. The string is now:

/065/017/132/137/133/033/135/

where the first two numbers are underlined to show that they now represent terminal symbols, i.e. indexes of concepts, while the others are still dictionary entries.

The word "INCLUDED" ($i = 132$) now refers us to element $k = 224$ in the table. This element tells us that, if 132 is followed by 137, then 137 should be destroyed, 132 should be replaced by 225 and we should go to the next word. On line 225 we find that 225 followed by 133 is 227 (erase 133). Then 227 followed by 033 is 228 (erase 033) and 228 followed by 134 would be 069, but we have 228 followed by 135 instead. Therefore, according to the algorithm, we go to the next line in the table, where we find that 228 followed by 135 is also 069. Since 069 is less than 150 we know that identification is complete. Entry no. 134 was the word "CATALOG" while 135 is the word "CATALOGUE": It is not surprising to find that they map into the same concept.

The input string has been reduced to a list of three integers—respectively the indexes of the descriptors: "COMPUTE-A-NUMBER", "STAR" and "IN THE CATALOGUE". These three numbers are all we need to determine the search strategy that will give the answer. To summarize, out input string has followed the transformations:

126	127	033	132	137	133	033	135
065		033	132	137	133	033	135
065		017	132	137	133	033	135
065		017	225		133	033	135
065		017	227			033	135
065		017	228				135
065		017	069				

Algorithms for logical translation

The analysis algorithm is clearly terminating. Therefore it has the effect of replacing the input string with a string of B^5 elements which is shorter. It is interesting to observe that, although we worked under Fortran, we were led to a list structure which brings to mind many features of the Information Processing Language V of NEWELL, SHAW and SIMON[2]. It will also be noted that lists such as B^4, B^5 and all our previously defined storage areas are susceptible to immediate generalization. It remains to be shown how an input string, after regularization of nonsubstantive terms and reduction to singular form, and after treatment by this algorithm, can be transformed into a retrieval formula.

Given our four-element signatures, let Q_j ($j = 1, 2, \ldots, 8$) be eight pushdown-stores (pds) of which four will be used to store images of signatures associated with a type n question, while all eight will be necessary to process a type p question. Let us denote by (w_i) the i-th concept on the list generated earlier. *The following algorithm will generate well-formed retrieval formulae* starting from the concept list:

Step (a) Check that w_1 specifies an instruction.
According to the rules of the formal language given above, the type of formula to be generated (p or n) may already be known to us. In general, however, decision must be postponed until further study of the context has taken place.

Step (b) Consider the next element and determine its meaning. In this system, the "meaning" of a descriptor is an integer m defined as follows: $m = 0$ if the concept plays the role of delimiter ("AMONG") or of instruction ("REQUEST"). $m = 1, 2, 3$ or 4 if the concept is either a technical entity ("KILOMETER") or an attribute ("SPECTROSCOPIC BINARY") *which refers to the m-th digit of the code.* For instance, the concept of "BINARY" has as its context the physical notion of stellar configuration, and it refers to the third digit of the code. Accordingly we would assign to it the meaning $m = 3$.

Step (c) If $m \neq 0$, load w_i into pushdown-store no. m and return to step (b).
If $m = 0$ and w_i is the concept "SUCH THAT", which we denote by the symbol $.\varepsilon.$ then block the pushdown-store where the preceding concept (w_{i-1}) has been loaded. This permits the detection of contradictions such as: "Find the number of single stars which are multiple". Return to step (b).
If $m = 0$ and w_i is the concept "AMONG" which we denote by the symbol $./.$ then note that the question is of type p. Block the first four pushdown-stores. In the following, concepts of meaning m will not be loaded into the pds of subscript m but of subscript $m+4$. Stay in step (c) until string terminates, then go to step (d).

Step (d) We now make a further check on the semantic validity of the question by reviewing the contents of the eight pds. In particular, if all stores are empty, the answer to the question is the total number of stars in the catalogue and no further processing is needed. Abortive action is taken, similarly, if the question is of type p and all stores are empty, which would be the case for a question such as: "What is the percentage of stellar systems among stars."

Step (e) Taking each pushdown-store in turn, a subprogram having access to the well-defined laws of the problem context (relationships between physical entities, definitions of the descriptors in terms of the code, etc.) *replaces the list of concepts by the actual symbols* used in the coding system. An "image" of the signatures to be retrieved is thus prepared. Note that the type of information stored in the eight pds switches from numeric to alphanumeric as a result of the execution of this step.

Step (f) Read the first symbol from the top of each Q_j ($j = 1,\ldots, 4$) and place parentheses around these four symbols. This is a logical unit. Continue step (f) until the first four pds are empty. Call E the expression obtained by placing the symbol of addition between these units. If E contains more than one term, nest it within parentheses. If the formula is of type n, go to step (g), else go to step (h).

Step (g) The formula: $V = E$ constitutes the solution.

Step (h) Go back to step (f) using now the last four pushdown-stores Q_j ($j = 5,\ldots, 8$). Call F the logical formula obtained. The formula: $V = E/F$ constitutes the solution.

Example: Consider the question

"What is the proportion of spectroscopic binaries among all main sequence systems which have only one component and whose total space velocity is less then fifty kilometers per second."

Subprogram A and the analysis algorithm reduce this sentence to the form:

"COMPUTE-A-FRACTION" "SPEC. BIN" $\boxed{./.}$ "DWARF" .∈. "SINGLE" .∈. "V" ">" "50"

Type p question

The above algorithm results in recognition of the question pattern and the loading of the descriptors into the pushdown-stores:

Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8
—	—	SP. BIN	—	DWARF	—	SINGLE	V
—	—	—	—	—	—	—	BELOW
—	—	—	—	—	—	—	FIFTY

At step (e) the descriptors are replaced by the actual symbols used in the coding system: And at step (h) we obtain the formula

$$V = (\,.\,.\,1\,.)/[(1{\cdot}01)+(1{\cdot}02)+(1{\cdot}03)+(1{\cdot}04)+(1{\cdot}05)+ \\ +(1{\cdot}11)+(1{\cdot}12)+(1{\cdot}13)+(1{\cdot}14)+(1{\cdot}15)].$$

This formula constitutes the output from the logical translator. It defines without ambiguity the search strategy to be executed by the searching automaton C.

Experiments with ALTAIR

We have just completed the description of a fully-implemented language processor. This program was written in Fortran and run on a CDC3400 computer. It has features in common with the question-answering program, BASEBALL, introduced in 1961 by researchers at the Lincoln Laboratory of M.I.T.[3]. In particular, the difficulty of the liguistic problem is comparable and the limitations placed on the input sentence are almost identical. The operators *and*, *or*, *not* are prohibited, with the exception that the operator *and* may be used in ALTAIR within the list of attributes of a certain population. Similarly, terms such as *most* or *highest* are prohibited. Such restrictions, as pointed out by the developers of BASEBALL, are minor ones and can easily be removed at a later stage of refinement of the technique.

Several experimental runs were conducted with ALTAIR over a period of 1 yr. The transcript of an actual question-and-answer exchange is given in the appendix. For purposes of illustration we have included some meaningless questions rejected by the program. The retrieval formulae are printed under valid questions, and the execution times have been tabulated (cumulative time, in minutes).

Like BASEBALL, our program has to deal with some problems of ambiguity in the assignment of meaning. For instance, in BASEBALL, some words have more than one meaning: "Boston" $\rightarrow$ *Place* = *Boston* or *Team* = *Red Sox*. In ALTAIR we find the same problem with words such as "between" or "higher than" for which appropriate meaning can only be selected by study of the context.

ALTAIR differs from BASEBALL in several important respects. First of all, our procedure for the assignment of meaning is based on a completely different scheme. Secondly, the introduction of the notion of *retrieval formulae*, the artificial language of the retrieval automaton, represents an effort on our part to formalize the design of question-answering systems beyond the idea of a *specification list* which was the fundamental concept in the structure of BASEBALL. In ALTAIR, it is true that we could visualize the pushdown-stores (Q_j) as playing the role of a set of specification lists, but the fundamental concept is found in the grouping of (Q_j) elements to constitute a retrieval formula. Many advantages are found in this formulation:

1. It becomes possible to handle *type* p *questions* both from a theoretical point of view and from the point of view of the programmer. Such questions are not allowed in BASEBALL. In ALTAIR the input sentence is not restricted to the form of a single-clause question.
2. The design of the entire program is simplified because the lower-order system (our automaton C) uses its own artificial language and thus becomes independent of the other systems. In particular, the efficiency of (C) will not suffer from alterations made in (A) or (B).
3. There is a one-to-one correspondence between well-formed retrieval formulae and meaningful interrogations of the data-base (i.e. in the case of ALTAIR, meaningful astronomical questions). Only knowledge of the data base coding system is necessary to decipher a retrieval formula. In particular, knowledge of the program structure itself is not required. Thus, testing of the system becomes an easy task, because the retrieval formulae corresponding to each question-and-answer pair can be checked visually without reference to the program:retrieval formulae have *physical meaning*.
4. By reliance on this structure, we obtain high efficiency in terms of speed. ALTAIR is capable of generating retrieval formulae from English sentences at the average rate of 250/min. The searching automaton is slower because it has to handle the characteristics of nearly nine thousand stellar systems. The execution time of the search is thus roughly proportional to the number of terms in the retrieval formula. In the experiments shown below, the average rate of processing was eight questions/minute.
5. Because of this structure, ALTAIR can be integrated as a unit into a larger system. The formalization of the intermediate artificial language of retrieval formulae opens the way to several types of optimizing schemes.

The fact remains that until the problem of self-optimization is solved, neither ALTAIR nor BASEBALL can be presented as the prototypes of a general solution to large-scale information retrieval. They represent only a first step towards such a solution.

APPENDIX

Q. Compute the number of systems having the HR region unknown.
$V = (0\ldots)$.

A. This number is 2298.

Q. What is the proportion of systems having stars?

A. 8933 stars out of 8933, or 100·0 per cent.

Q. Request all stars redder than $F8$ among main sequence systems.

A. Question not compatible with current coding system and/or semantically incorrect.

Q. Request all main sequence systems redder than $F8$.
$Y = [(1G..)+(1H..)+(1I..)+(1J..)+(1K..)+(1L..)]$.

A. This number is 331.

Q. Request all stars between $F3$ and $F7$ having speed above forty.
$Y = [(.F.5)+(.F.6)+(.F.7)+(.F.8)+(.F.9)]$.

A. This number is 80.

Q. What is the proportion of stars having a speed above 40 km/sec among the systems whose spectral type is between $F3$ and $F7$.
$V = [(\ldots5)+(\ldots6)+(\ldots7)+(\ldots8)+(\ldots9)]/(.F..)$.

A. 80 stars out of 534, or 15·0 per cent.

Q. Compute the number of stars of MK class III whose spectral type falls between $F3$ and $K2$.
$V = [(2F..)+(2G\ldots)+(2H..)+(2I..)]$.

A. This number is 1463.

Q. What is the proportion of sub-giants among those stars which are of spectral type earlier than $F7$?
$V = (4\ldots)/[(.A..)+(.B..)+(.C..)+(.D..)+(.E..)+(.F..)]$.

A. 333 stars out of 4812, or 6·9 per cent.

Q. How many systems of M-K Class VI are included in the bright star catalogue?

A. Question not compatible with current coding system and/or semantically incorrect.

Q. How many stellar systems have primaries?
$V = (\ldots.)$.

A. This number is 8933.

Q. What is the proportion of systems having a velocity in excess of 80 km/s among all stars which have a common proper motion companion?
$V = (\ldots9)/[(..6.)+(..7.)+(..8.)+(..9.)]$.

A. 8 stars out of 428, or 1·9 per cent.

Q. Find the percentage of visual pairs among all stellar systems where the primary is redder than $F3$ and whose velocity is below 20 km/sec.

A. Retrieval formula length exceeded.

Q. Compute the number of stars of MK Class III whose spectral types fall between Type $F3$ and Type $K2$.
$V = [(2F..)+(2G..)+(2H..)+(2I..)]$.

A. This number is 1463.

Q. Calculate the cardinality of the set of all giants which are bluer than Type $K2$.
$V = [(2A..)+(2B..)+(2C..)+(2D..)+(2E..)+(2F..)+(2G..)+(2H..)+(2I..)]$.

A. This number is 1744.

Q. Bill has come up with the following questions.

A. The following word is unknown..Bill.

Q. Request common proper motion stars which are bluer than giants.

A. Question not compatible with current coding system and/or semantically incorrect.

Q. Find the number of double spectroscopic binaries.
$V = (..5.)$.

A. This number is 16.

Q. How many subgiants have a common space velocity?

A. Question not compatible with current coding system and/or semantically incorrect.

Q. Request all giants having common proper motion components.
$Y = [(2.6.)+(2.7.)+(2.8.)+(2.9.)]$.

A. This number is 119.

Q. How many giants are double?
$V = [(2.2.)+(2.3.)+(2.4.)+(2.5.)+(2.6.)+(2.7.)+(2.8.)+(2.9.)+(2.T.)]$.

A. This number is 328.

Q. How many subgiants are double?
$V = [(4.2.)+(4.3.)+(4.4.)+(4.5.)+(4.6.)+(4.7.)+(4.8.)+(4.9.)+(4.T.)]$.

A. This number is 133.

Q. Request all dwarfs having common proper motion components.
$Y = [(1.6.)+(1.7.)+(1.8.)+(1.9.)]$.

A. This number is 175.

Q. What is the number of stars?

A. This number is 8933.

Q. How many giants are in the catalogue?
$V = (2...)$.

A. This number is 2644.

Q. How many subgiants are included in the Bright Star Catalog?
$V = (4...)$.

A. This number is 541.

Q. Find the number of stars that are binaries and giants.
$V = [(2.2.)+(2.3.)+(2.4.)+(2.5.)+(2.6.)+(2.7.)+(2.8.)+(2.9.)+(2.T.)]$.

A. This number is 328.

Q. How many stars have space velocities in excess of 20 km/s?
$V = [(\ldots 3)+(\ldots 4)+(\ldots 5)+(\ldots 6)+(\ldots 7)+(\ldots 8)+(\ldots 9)]$.

A. This number is 3077.

Q. What is the color excess of $F3$ stars?

A. Question not compatible with current coding system and/or semantically incorrect.

Q. How many numbers fall between 20 and 30?

A. Question not compatible with current coding system and/or semantically incorrect.

Q. How many binary stars are multiple?
$V = [(..2.)+(..3.)+(..4.)+(..5.)+(..6.)+(..7.)+(..8.)+(..9.)+(..T.)]$.

A. This number is 1513.

Q. Find the number of stars between $K7$ and $B3$.
$V = [(.B..)+(.C..)+(.D..)+(.E..)+(.F..)+(.G..)+(.H..)+(.I..)+(.J..)]$.

A. This number is 8069.

Q. Find the number of stars having space velocities between 20 and 40 km/s.
$V = [(\ldots 3)+(\ldots 4)]$.

A. This number is 2013.

Q. How many stars are double dwarfs?
$V = [(1.2.)+(1.3.)+(1.4.)+(1.5.)+(1.6.)+(1.7.)+(1.8.)+(1.9.)+(1.T.)]$.

A. This number is 761.

Q. How many systems are included in the Color Luminosity Diagram?

A. Question not compatible with current coding system and/or semantically incorrect.

Q. How many double stars have common proper motion companions?
$V = [(..6.)+(..7.)+(..8.)+(..9.)]$.

A. This number is 428.

Q. What is the number of dwarfs between $B3$ and $B7$?
$V = (1B..)$.

A. This number is 312.

Q. How many stars are redder than $K8$ but are dwarfs?
$V = [(1K..)+(1L..)]$.

A. This number is 1.

Q. What is the proportion of spectroscopic binaries among all main sequence systems which have only one component and whose total space velocity is less than 50 km/sec.
$V = (..1.)/[(1.01)+(1.12)+(1.03)+(1.14)+(1.05)+(1.11)+(1.02)+(1.13)$
$+(1.04)+(1.15)]$.

A. 545 stars out of 1722, or 31·6 per cent.

REFERENCES

[1] J. VALLEE and J. A. HYNEK: An automatic question-answering system for stellar astronomy, *Publ. Astron. Soc. of Pacific*, 1966, 78, 315.

[2] A. NEWELL and F. TONGE: An introduction to information processing language V, *Comm. A.C.M.*, 1960, 3, [2], 205–211.

[3] B. GREEN, A. WOLF, C. CHOMSKY and K. LAUGHERY: BASEBALL: an automatic question-answerer, *Proc. of the WJCC*, 1961, 219–224.